SPACE SCIENCES LABORATORY

UNPUBLISHED PRELIMINARY DATA

UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA

A UNIVERSAL ADAPTIVE CODE FOR OPTIMIZATION (GROPE)

by

Merrill M. Flood

and

Alberto Leon

Internal Working Paper No. 19

August 1964

Space Sciences Laboratory
University of California
Berkeley

This research was supported in part by the National Aeronautics and Space Administration under General Grant #NSG-243-62 under the University of California. [redacted]

CONTENTS

INTRODUCTION[*]

A UNIVERSAL ADAPTIVE CODE FOR OPTIMIZATION, for us, is a general flexible computer program for numerical analysis which selects adaptively and sequentially among a group of several optimizing subroutines as each problem calculation progresses. The optimizing subroutines are selected and controlled by the so-called executive part of the program. Each optimizing subroutine is selected probabilistically at each stage of the solution of any one problem. The selection probabilities are based upon the past history of each optimizing subroutine.

The task here, called the optimization process, is to locate a set of values of a set of variables that yields the optimum value for a function given algebraically. The function to be optimized may be constrained or unconstrained.

The UNIVERSAL CODE (GROPE) is intended to serve three different purposes in different and independent phases of the optimization process.

The first purpose is the efficient optimization of a given mathematical function (the function to be optimized, either by minimization or by maximization). Here the solution process will be accelerated by the action of the optional features present in each one of the optimizing subroutines. This combination of procedures is expected to yield a solution more quickly than would be obtained by using any one of the subroutines by itself.

The organization of the computing system is the second purpose. It is only possible to have in core memory a limited number of optimizing

---

[*]The initial steps toward a Universal Adaptive Code for Optimization (GROPE) were described in Internal Working Paper No. 11, Space Sciences Laboratory, April 1964, by Alberto Leon.

subroutines at any one stage of the calculation, the number at any time depending also upon their size. It is obviously desirable to have in high-speed core memory those subroutines having the highest usage rate. Such selection eliminates much of the swapping back and forth between higher and lower speed memories within the configuration, thus considerably increasing the computing system's efficiency.

The relative effectiveness of an optimizing subroutine depends at least partially upon the properties of the mathematical function to be optimized. It is expected that problems arising from a particular population of users will have similar properties. There are situations in which it is possible to identify a particular class of problems as currently predominant among the users. A third purpose is adaptive automatic selection of currently effective subroutines, as indicated by recent users' experience.

We have selected a group of iterative optimizing subroutines to build up the UNIVERSAL ADAPTIVE CODE. The iterative techniques selected have certain similarities that make possible sequential comparisons among them as to their current effectiveness, thus providing the essential basis for adaptive selection among them during a calculation.

A linear stochastic learning model used for other purposes by one of the authors[1] provides the adaptive mechanism of the code. It is very simple to use any other such adaptive mechanism for this portion of the code, and several others will be tested under our present project; this requires merely that subroutines ADAPT and RCPRO be rewritten so as to represent the adaptive mechanism to be used.

---

[1]Flood, Merrill M., "Stochastic Learning Theory Applied to Choice Experiments with Rats, Dogs and Men," Behavioral Science, 1962, 7, pp. 289-314.

GROPE is available in FORTRAN IV for IBM 7090/7094 computers processed by the FORTRAN IV compiler, a 7090/7094 IBJOB Processor Component. The code operates under the University of California (Berkeley) Executive System. Some of the optimizing subroutines were translated from FORTRAN II into FORTRAN IV using SIFT (Share Internal Fortran Translator) which automatically translates a FORTRAN II source program (or subprogram) into a FORTRAN IV source program.

The remainder of this paper is devoted to detailed descriptions of various portions of the UNIVERSAL CODE. First a discussion of the structure of the Main Program. The Main Program contains subroutines which: (1) initialize the process, by identifying subroutines, parameters and other pertinent information, (2) select optimizing subroutines for use together with relevant information for the particular subroutine, (3) stop the optimizing subroutine in use based upon stopping criteria, (4) evaluate the optimizing subroutine's effectiveness, (5) modify the selection probabilities for each optimizing subroutine based upon evaluated effectiveness, and (6) stop the optimization process and provide the necessary print out. The subroutines are explained in the order in which they appear for the first time in the program. The Flow Charts in Appendix C should be helpful in following the description.

Following the discussion of the general structure of the program are descriptions and comments regarding specific optimizing subroutines used in past operations and a description of the overlay feature for core storage utilization. Finally some results of past operations are discussed together with remarks about flexibility and limitations of the program.

STRUCTURE OF GROPE

A brief explanation of the structure of the UNIVERSAL ADAPTIVE CODE, hereafter called GROPE, is given in this section. Descriptions of the subroutines are also given here together with some of the most important variables. The names chosen for the subroutines and variables are mnemonics, to help make the description easier to follow and remember.

A simplified Flow Chart of GROPE is included in Appendix C. Complete FORTRAN IV listings and a sample of data are given in Appendix D.

The general code, as it is now, has some subroutines that the user must write and compile in order to treat his particular optimization problems. However, a separate run is needed for each problem if he has different optimization functions to treat. In the description of subroutines that follows, this feature will be clarified.

MAIN PROGRAM

The MAIN program is just a sequence of calls to the proper subroutines. MAIN includes also the initialization of certain control quantities, the clock readings for control purposes, and the printing of the final output. One control key* (KR8), read in as data, controls the number of problems to be run and another one the amount of information to be printed as final output.

*A description of the effect of each control key may be found in Appendix B.

The final output includes:

a. Identification of the problem.

b. Best functional value (PFMAX), together with the optimizing vector (PBESV(J)).* In cases of linear problems (i.e., linear programming problems or systems of linear equations) the vector of residuals (TEMPE(J)) is also a part of the final output.

c. Number of the trial** at which the optimum point was reached (IBEST) as well as the number of the optimizing subroutine (IROUT) in use at that time.

d. Total number of calls to optimizing subroutines (NLOOK) and distribution of these calls among the different subroutines (ISH(I)).

e. Total elapsed time for each problem (NTIME) and total number of functional evaluations (NEL1).

f. Total time spent by each optimizing subroutine (NT(I)) in each independent problem.

g. Vector to initialize the random number generator.

Whenever KR 14 is set greater than zero the following additional information is printed:

h. The final state (vector of probabilities) of the adaptive mechanism represented in ADAPT (PROB(I)).

i. The vector with the number of the optimizing subroutine used in each trial (JRES(L)).

*We use consistently the following subscripts as mnemonics: I, it ranges from 1 to the available number of codes (NCODS); J, it ranges from 1 to the total number of variables (N); L, it ranges from 1 to the total number of trials (MXTRI).

**Our definition of trial is given in the description of subroutine ADAPT.

j. A vector of 0's and 1's (JSTI(L)). A zero indicates a failure of the optimizing subroutine used in that trial and a 1 denotes a successful use.

If several problems were solved in the run and the final output also shows the total elapsed time for all of them and the total number of problems. This case only arises when several problems are solved for functions differing only in the sets of defining parameters.

I. Subroutine RWSUB

The first subroutine called by MAIN is RWSUB.

RWSUB is used to read from DATA, and then print, the names of the subroutines used, identified also by the date of their compilation and any other information pertaining to each subroutine used in the run.

This is just a way of recording identification information in the printed output to facilitate analysis of results, debugging and testing matters.

Any other Hollerith statements or comments, as desired, may be read from DATA and then printed in the output using RWSUB in the same manner.

II. Subroutine RSDTA

RSDTA reads input data. It reads in all the necessary parameters for the optimizing subroutines together with the ones controlling the general work of the code. Values for the following variables and parameters are read in:

KR, KR1, ..., KR25 denote control keys

N denotes the number of variables, and in linear problems N is normally one more than the number of unknowns because the homogeneous form is used.

M denotes the number of equations or inequations in linear problems. It is also available for other uses when linear problems are not being treated.

NC is normally set equal to N. NC will be used in BATCHE, a subroutine not included yet in the code.

NTSNC denotes the maximum number of times subroutine SNICON can be used.

NTC denotes the maximum number of variables to be analyzed. In linear problems it is normally one less than N.

NORVI(J) denotes the order in which the variables will be analyzed. Recombination of NORVI(J) and NTC gives flexibility for analyzing the variables in any desired order.

MXTRI denotes the maximum number of trials to be performed. MXTRI is reached unless failure of all the iterative procedures occurs in sequence or NTSNC is attained.

NDECI denotes the number identifying the chosen criteria for ending trials (e.g., a certain number of functional evaluations).

MXNEL denotes the maximum number of functional evaluations allowed within a trial.

MXTIM denotes the maximum amount of time, in some convenient unit, allowed for any one trial.

IMDS denotes the number identifying the adaptive mechanism used in ADAPT whenever several are available.

LL1 is a control parameter used in subroutine RCPRO. LL1 must be set equal to unity at the start of each run, and it is modified by the program at later stages.

LL2 is a control parameter used by subroutine EVALT. LL2 must be equal to unity at the beginning of each run.

LL3 is a control parameter used by subroutine OUTPUT. LL3 must be set equal to unity at the beginning of each run.

LL4 is a control parameter used by subroutine OPTIMU. LL4 must be set equal to unity at the beginning of each run.

LL5 is a control parameter used by subroutine SNICON. Its role is explained in the description of subroutine SNICON.

LL6 is a control parameter used by subroutine ADAPT. LL6 must be set equal to unity at the beginning of each run.

LOR is a control parameter used by the optimizing subroutines. LOR must be set equal to unity at the beginning of each run.

EP1 denotes the parameter ϵ_1 that is used to define when the functional value at the end of one trial, FMAX, is "better" than the functional value at the end of the preceding trial, PFMAX.

TOL denotes the fraction of acceptance for improvement in functional value between two consecutive cycles of one optimizing subroutine.

TOLS denotes the smallest error accepted in the solution of systems of linear equations.

IN(K) denotes a three integer element vector used to initialize the pseudo-random number generator whenever called by subroutine ADAPT.

IN1(K)denotes a three integer element vector used to initialize the pseudo-random number generator whenever called by subroutine TOSS.

IN2(K), IN3(K) denote three integer element vectors used to initialize the pseudo-random number generators. These are free to be used in any subroutine written by the user and associated with his problem.

PN(J) denotes the starting vector.

PNMAX(J) and PNMIN(J) denote upper and lower limits for each variable.

NCODS denotes the number of optimizing subroutines available.

The following is a set of parameters used by the optimizing subroutine RANDOM:

MCYCLE denotes the maximum number of cycles the subroutine can be used. It is reached unless one of the internal stopping rules of the subroutine is attained first.

NTES denotes the maximum number of functional evaluations within a cycle.

NTNCYS denotes the maximum number of continuous cycles accepted without significant improvement in the functional value.

The following are the parameters used by the optimizing subroutine SHRINK:

MCYCLO denotes the total number of cycles to be performed. It is attained unless one of the internal stopping rules is reached first.

NTOS denotes the maximum number of functional evaluations per cycle.

NTNCOS denotes the maximum number of continuous cycles accepted without significant improvement in the functional value.

SHRON denotes the fraction by which the operating space is reduced after every complete cycle.

The following set of parameters is used by the optimizing subroutine SATTER:

MRDTN denotes the total number of cycles to be performed. It is reached unless one of the internal stopping rules is attained first.

NTIS denotes the total number of functional evaluations to be performed in a cycle.

NDTN denotes the maximum number of continuous changes in direction accepted without significant improvement in the functional value.

STEP denotes the step size.

The following is a list of the parameters utilized by optimizing subroutine LOOK:

DEL denotes quantity specifying incremental change in all variables.

DELR denotes ratio by which DEL is to be changed. DELR is always equal to or less than unity.

DELMIN denotes the minimum allowable value of DEL.

DELTA denotes a quantity specifying size of pattern move.

SHRUN denotes an adjusting factor for DELTA.

The following variables are used by subroutine SNICON:

DAMBL denotes the relaxation parameter in linear and nonlinear programming problems, and is available for other uses in other problems.

THETA denotes an exponential reduction factor θ, taking DELMIN to θDELMIN when SNICON is called more than once in a linear problem. THETA is free for other uses in other problems.

OMEGA plays the same role for SHRUN that THETA plays for DELTA; again for linear problems and free for other uses otherwise.

TAU denotes the adjusting factor for the relaxation parameter DAMBL, calculated for mathematical programming problems as a function of the actual error and the maximum permissible error; free for other uses in other problems.

RO denotes the adjusting factor for TAU in mathematical programming problems, making TAU greater after each adjustment of the relaxation parameter; free for other uses in other problems.

RISTK denotes the maximum allowable error in satisfying the linear inequalities in mathematical programming problems; free to be used otherwise in other problems.

KR14 controls the printing of the information previously read in as data. If KR14 is less than or equal to zero then printing is executed; otherwise control is returned to MAIN.

A. Subroutine RFMTS

RFMTS makes it possible to read formats as part of the data, to a maximum of ten different formats. These variable formats are left available to the users. KR15, in subroutine RSDTA, controls the calling of RFMTS. If KR15 is greater than zero then RFMTS is called. Otherwise transfer is made to the reading statements of RSDTA.

III. The Executive Portion of the Program

It is intended here to present what we consider the most important portion of the program: the adaptive process. A brief introduction with the role of each one of the subroutines involved is included first together with a general description of the process. This introduction is followed by a more detailed presentation of subroutine ADAPT and subroutine OPTIMU. The work of other related subroutines, e.g., DECIS, is described as it appears convenient.

A. Overview

Subroutine ADAPT is the adaptive mechanism portion of the program. It selects one from the group of optimizing subroutines for use in each trial and decides whether the most recent use of each is to be considered a SUCCESS or a FAILURE.

Subroutine ADAPT delegates many of its functions to subroutines LPRMT, IDECI, OPTIMU, DECI (through OPTIMU), OUTPUT and RCPRO. Each of these six subroutines is described briefly in subsequent paragraphs of this section. A simplified Flow Chart of ADAPT is included in Appendix C.

Subroutine LPRMT reads in both the structure and the state (initial probabilities) parameters of the adaptive mechanism ADAPT.

Subroutine RCPRO recomputes the state probabilities after every trial. Here a trial is defined as the use of some one optimizing subroutine for a specified period of time, as for some maximum number of functional evaluations, or in accordance with some other selected criterion. At the end of each trial the functional value is compared against the best previous one. If the present functional value is the best, the result of the trial is considered a SUCCESS and otherwise a FAILURE.

The information concerning the technique in use, the trial number, and the condition at the end of the trial (i.e., SUCCESS or FAILURE) is utilized by RCPRO to recompute the new state probabilities of the system. The optimizing subroutine with the highest number of previous successes, after several trials, will formally have the highest probability of reuse.

Subroutine OPTIMU calls the different optimizing subroutines in accordance with the decisions made by ADAPT. OPTIMU also keeps account of the number of times each optimizing subroutine is called, as well as the amount of time each one of them is used. Subroutine EVALT, the one carrying out the functional evaluation whenever required, is called by OPTIMU; counts of the number of calls to EVALT are stored as NEL and NEL1. A simplified Flow Chart of subroutine OPTIMU is included in Appendix C.

Subroutine IDECI initializes the decision criterion at the beginning of each trial. These criteria are used later by subroutine DECIS, called by OPTIMU, to decide whether or not the end of a trial has been reached.

Subroutine OUTPUT is controlled by one of the control keys. Whenever OUTPUT is called, either by ADAPT or OPTIMU, most of the information available at the calling time is printed out. The information may contain: the number of the optimizing subroutine currently in use, the actual state probabilities of the system, values of variables, functional value, best previous functional value together with the corresponding vector, number of functional evaluations within the trial, accumulated number of functional evaluations, and so forth.

B. Subroutine ADAPT

It is convenient to describe in detail the operation of ADAPT together with the related subroutines. Definition of the variables will be helpful also.

The following variables will be found in the FORTRAN IV listings included in Appendix D:

I1 denotes the number of times the optimizing subroutines have been unsuccessful in sequence. Whenever I1 reaches a value equal to the total number of optimizing subroutines (NCODS), this means that all FAILED in sequence.

J1 denotes a counting parameter ranging from unity to NCODS, and is used to indicate the number of the optimizing subroutine selected for next use. Once the selection is made ICODE is set equal to J1.

ICODE denotes the number of the optimizing subroutine in use at any given time.

ITRIA denotes the number of the trial.

IBEST denotes the number of the trial at which the optimum functional value was attained. Each time a better functional value is found IBEST is set equal to the number of the trial at that moment.

IROUT denotes the number of the optimizing subroutine in use during trial numbered IBEST.

KEY(I) denotes the condition causing return from the Ith optimizing subroutine to OPTIMU. If KEY(I) is unity this indicates normal return; to call EVALT, check decision criteria etc. If KEY(I) is zero, this indicates that one internal stopping rule was reached.

LIZ(I) denotes the number of times optimizing subroutine numbered I is called in sequence, after failure in use, and before another subroutine is a success. If LIZ(I) is unity, then I1 is increased by unity, and if greater than 1 the control is transferred to select a new optimizing subroutine.

LOCK(I) records success or failure of the optimizing subroutine numbered I for its most recent use: success if LOCK(I) = 1 and failure if

LOCK(I) = 0.

The first thing done by subroutine ADAPT, whenever LL6 = 1, is to call subroutine LPRMT. LL6 is read in by RSDTA, where it is set equal to unity. LL6 is set equal to two immediately after returning from LPRMT so in subsequent calls to ADAPT the statement calling LPRMT will be skipped.

As was said previously, the state probabilities of the system are recomputed at the end of every trial by RCPRO. The code presented here for ADAPT, and for RCPRO, makes use of a linear stochastic learning model as the adaptive mechanism; RCPRO is easily rewritten to represent any alternative adaptive mechanism that may be of interest, and several such mechanisms can be included as options within GROPE if so desired.

The information required by the symmetry learning model, as coded in our present version of ADAPT, is as follows:

ITRIA, the number of the present trial

$PROB_I$(ITRIA-1), the vector of the state probabilities from the previous trial, where I is ranging from 1 to NCODS

JRES(ITRIA), the number of the subroutine used in the present trial

JSTI(ITRIA), set at zero after a failure and at unity after a success.

The output of RCPRO will be a revised value for $PROB_I$(ITRIA), representing the new state of the system after trial numbered ITRIA.

After $PROB_I$(ITRIA) has been calculated following trial ITRIA, ITRIA is increased by unity and the result compared against the maximum number of trials to be performed. If the maximum number of trials has been reached control is returned to MAIN where final output is printed out.

If there are more trials to go transfer is made to begin a new trial by closing loop $\textcircled{\Omega} \longrightarrow \textcircled{\Omega}$.

Whenever transfer is made to close loop $\textcircled{\Omega} \rightarrow \textcircled{\Omega}$ the counting parameter J1 is initialized to unity. It is numbering the order in which the optimizing subroutines are called. A set of random numbers between zero and unity is generated next. The set contains a total of NCODS random numbers. The element J1 of the current vector of state probabilities is compared against the corresponding element of the vector of random numbers in statement number 6 of ADAPT. If $PROB_{J1}(ITRIA) > RNB(J1)$* the optimizing subroutine numbered J1 is chosen to be used and so

$$ICODE = J1$$

If $PROB_{J1}(ITRIA) \leq RNB(J1)$, J1 is increased by unity and the previous test is performed again. Whenever J1 reaches a value equal to NCODS with no selection of optimizing subroutine transfer is made to statement 7 to generate a new set of random numbers, initialize J1 to unity and to compare the vector PROB against the vector RNB. A suitable value of RNB will be eventually found so as to permit the selection of a member of the group of optimizing subroutines.

Once the selection of optimizing subroutine has been made, the element LOCK(ICODE) of the vector LOCK is compared against zero. The vector LOCK(I) is set equal to unity, for I = 1, NCODS, at the start of ADAPT as well as each time a subroutine is SUCCESSFUL. LOCK(ICODE) is changed to zero whenever subroutine numbered ICODE fails during the calculation process.

The role the vector LOCK(I) plays is very important, for it precludes the reuse of a particular optimizing subroutine immediately after

---

*RNB stands for random number.

failing but also before at least one other subroutine has been a success in its last use. In other words, LOCK(I) blocks the possibility of having a repeated use of a subroutine under conditions where it had previously failed. Each trial when an optimizing subroutine succeeds the corresponding value of LOCK(I) is again set equal to unity.

If LOCK(ICODE) is zero then transfer is made to statement 8 where LIZ(ICODE) is incremented by unity. Next, LIZ(ICODE) is compared against unity. If LIZ(ICODE) = 1 then the subroutine numbered ICODE is called after a failure and I1 is incremented by unity in statement 160. The next step is to compare I1 against NCODS. Whenever I1 > NCODS then all the optimizing subroutines have failed in sequence, statement 10 is printed out indicating this fact, and control is returned to MAIN. When LIZ(ICODE) > 1 this indicates that more than one attempt to call the subroutine numbered ICODE has been made after failure and before any other subroutine has succeeded; hence control is transferred to statement 4 to select another optimizing subroutine.

LIZ(ICODE) is then ensuring that I1 is increased by unity only when subroutine numbered ICODE is chosen for the first time following unsuccessful use. It can be seen that any optimizing subroutine may be selected at any time because of the selection process already described. It is required, consequently, with this selection process, to have the control vectors LOCK(I) and LIZ(I) as explained.

The NO branch of the logical statement IS LOCK(ICODE) = 0 ? transfers control to statement 9. I1 and the vector LIZ are again initialized, and the vector JRES of responses is updated in statement

JRES(ITRIA) = ICODE .

Subroutine IDECI is next called to initialize the chosen decision criteria. For example, if the deciding criteria are numbers of functional evaluations then NEL is set equal to zero. NEL records the number of functional evaluations within a trial, and NEL(1) records the total number of functional evaluations made so far. The control parameter NDECI, read in as DATA, denotes the number identifying the chosen decision criteria and as such guides the operation of subroutines IDECI and DECI.

After the decision criteria have been initialized subroutine OPTIMU is called. The operational aspects of OPTIMU will be described briefly in one of the later parts of this section. Immediately after returning from OPTIMU, at the end of one trial, the result of the trial is defined. The functional value at the end of one trial, FMAX, is "better" than the functional value at the end of the preceding trial, PFMAX, if and only if

$$\mathrm{PFMAX} - \mathrm{FMAX} > \mathrm{EP1}$$

If this condition is met then control is transferred next to statement 20 to build up the stimuli vector, to store FMAX in the location of PFMAX, to record the trial's number (IBEST = ITRIA), to store the variable's vector as PBESV(J) and to reset LOCK to unity. The next program command is

$$\mathrm{LOCK(ICODE)} = \mathrm{KEY(ICODE)}\ .$$

The elements of the vector KEY were set initially equal to unity. The element ICODE of the vector KEY is changed to zero whenever the optimizing subroutine numbered ICODE reaches one of its internal stopping rules. KEY is intended to block a new call to subroutine numbered ICODE in a stage of the calculation process where it reached some internal stopping rule, e.g., lack of convergence after certain maximum number of iterations. The role played by KEY is very similar to that of LOCK. KEY and LOCK are interrelated and it is important to remember that reaching one internal stopping rule of the optimizing subroutine does not mean unsuccessful use; the work of the subroutine, up to that point, may be very well better than the previous one. Nevertheless, and because the control on calls is exercised by LOCK, it is required to have the substitution statement described previously. Hence, LOCK(ICODE) will present a value of zero even though subroutine numbered ICODE was not considered a failure by ADAPT.

A comment is printed out now indicating the number of the trial, the optimizing subroutine used, the fact that the use of the subroutine has been a success and the number of functional evaluations.

In such cases where the result of the trial is not considered better than the previous one control is transferred next to statement 21. The bookkeeping operations described for the successful use are carried out here for the unsuccessful one, that is,

```
JSTI(ITRIA) = 0
LOCK(ICODE) = 0
PRINT OUT COMMENT
```

Subroutine OUTPUT may be called here under the discretion of KR5. After returning from OUTPUT or after skipping the command calling OUTPUT, the state probabilities of the system are recomputed by subroutine RCPRO, the number of trials increased by unity and finally loop $\textcircled{\Omega} \longrightarrow \textcircled{\Omega}$ is closed as described earlier in this section.

It is worthwhile to explain briefly the operation of subroutine OPTIMU since it plays an important role in the operation of ADAPT.

C. Subroutine OPTIMU

After one of the optimizing subroutines has been selected, and provided previous initialization of the decision criterion carried out by subroutine IDECI, subroutine OPTIMU is called by ADAPT.

The program parameter LL4 determines the entry point to OPTIMU. LL4 is set equal to unity by RSDTA so the entry point for the first time will be statement 9. The vectors LE(I) and LUT(I) are initialized to unity and zero respectively. The role of LE(I) and LUT(I) is found in a subsequent section describing the optimizing subroutines. Subroutine EVALT is called next to evaluate the objective function at the initial point. The starting point must be selected by the experimenter and it is obvious that the chosen point has an important effect upon the amount of time necessary to complete the optimization computation. Knowledge of the experimenter concerning the problem under study will be of great value in selecting a point not too distant from the one he is seeking. After the function has been evaluated at the starting point the following substitution statements are executed

```
FMAX = SN
PFMAX = SN
PBESV(J) = PN(J)
BESV(J) = PN(J) for J = 1, N
```

The functional value at any time is stored as SN and the corresponding vector as PN. The substitution statements are just the initialization of those variables to the best values so far, that is, the initial ones.

LL4 is reset equal to two and it keeps this value unless modified by subroutine SNICON in later stages of the process. Hence, subsequent calls to subroutine OPTIMU will have as entry point statement 10.

The portion of subroutine OPTIMU between statements 10 and 20 contains the commands to call the different optimizing subroutines. This portion contains also clock readings and counting parameters to record the time each one of the subroutines is used and the number of times they were called respectively. NLOOK stores the number of times the optimizing routines are used all together.

The next statement is a computed GO TO guided by NGO as follows:

```
GO TO (21, 25), NGO
```

NGO brings its value from the optimizing subroutine in use. NGO is set equal to two if some internal stopping rule of the subroutine occurs. Here, a comment is printed out indicating the number of the subroutine, the number of the trial and the number of functional evaluations in the trial. The element ICODE of the vector KEY is set equal to zero as was previously explained, the parameter LOR equal to unity (its work is described later) and the vector LE equal to unity. Control is then returned to ADAPT for the selection of a new subroutine.

Each time the optimizing subroutines require a functional evaluation control must be transferred to OPTIMU which calls subroutine EVALT. EVALT evaluates the function being optimized; in this sense, EVALT defines the optimization function. NGO is set equal to unity by the subroutine in use and transfer is made to statement 21 rather than to statement 25.

Subroutine DECIS is called in statement 21. DECIS merely tests if the chosen decision criterion for switching from one optimizing subroutine to another (end of one trial) has been met and if so it sets LGO equal to two, control is transferred next to statement 14 where LOR and the vector LE are reset to unity before returning to ADAPT. If the decision criterion has not yet been reached LGO is set equal to unity and control transferred to statement 15. Subroutine EVALT is called at this point, the control parameters NEL and NELI are increased by unity and subroutine OUTPUT called if KR20 is so indicating. Transfer is made next to statement 10 where the same subroutine is evoked again and the whole process reinitiated.

Before finishing this section on the adaptive process it is of interest to include some comments suggested by our limited experience with the program.

D. Some Comments on the Adaptive Process

There are many obscure aspects in relation to the work of ADAPT and OPTIMU and their related subroutines. New ideas, changes, deletions, and other suggestions of readers will be most welcome.

Perhaps the most important point and, unfortunately, the most obscure one, concerns the proper time for switching from one optimizing subroutine to another. On the one hand, the criteria should be general enough to include measurable and definite points compatible with all the subroutines (number of functional evaluations, periods of time of equal duration at work, etc.). On the other hand, basing the changes upon too general criteria is dangerous. By means of general criteria we can perfectly well be calling a new subroutine precisely at the moment in which the present subroutine as a consequence of a set of just exploratory attempts is after the right path. This is extremely important if we wish to introduce accelerating convergence procedures.

The comments of the preceding paragraph strongly suggest a homogeneous group of subroutines, with similar iterative structure even though conceptually different. The selected subroutines, for example, should be able to begin the computation process at any arbitrary point. That point could be the experimenter's decision at the very beginning of the process or the last point left by another subroutine as a consequence of the switching criteria.

Accelerating convergence procedures are of great importance and valuable help. It is interesting to say that while it is important to work with a homogeneous set of optimizing subroutines we should not forget that proper combinations of apparently nonhomogeneous techniques may speed up the convergence to an answer.

Another important point has to do with the effect of the sequence of calling optimizing subroutines upon the efficiency of the optimization process. Identical values shall be assigned to the initial state probabilities whenever there is no information on hand about the problem. As a consequence each optimizing subroutine has the same probability to be selected at the beginning of the calculation in accordance with the random selection already described. In accordance again with the selection process presented, the optimizing subroutine with the highest probability will normally have the highest chance to be reused in later stages of the calculation. We believe this to be a satisfactory way to select optimizing subroutines. However, we are somewhat concerned about the effects of the calling sequence. Later research may lead to better ways of selection.

IV. Subroutine SNICON

SNICON is called by MAIN, whenever KR16 > 0, after an optimization calculation is completed. SNICON provides a means for repeating an optimization calculation on the same problem with altered parameter values, and possibly also with new values for the optimization parameters. Each recalculation is completely independent of the preceding ones, and is normally used to check previous solutions.

One standard option (LL5 = 1) starts the recalculation at an initial point remote from the optimum point already located, but within the region defined by the constraints (PNMAX and PNMIN). If the variables are unconstrained, the user must provide arbitrary constraints that will then be used to determine the location of the remote point.

The recalculation can be arranged as a continuation of the preceding one (LL5 = 2). In this instance, the new initial point will normally be at the optimum already found. The usual objective in this is to permit change in the search parameters to provide increased accuracy in the results. In problems involving relaxation parameters that require readjustment during the iterative optimization process, as for mathematical programming problems, SNICON does this.

The control parameter NTSNC, as was described in subroutine RSDTA, specifies the number of recalculations to be made under SNICON.

OPTIMIZING SUBROUTINES

Four subroutines were chosen to form our first set of optimizing procedures for GROPE which are briefly described in this section.

The direct search technique known as BEST UNIVAR developed by the authors[2, 3] at The University of Michigan is ready to be added to GROPE. Four more subroutines are under study to introduce the necessary changes to make them compatible with GROPE. These subroutines are VARMINT, STEP, MINFUN and PARTAN. Some experiences with these optimizing procedures are reported elsewhere.[4]

The four optimizing subroutines forming the core of GROPE were written as subroutines of our general master programs for optimization (part of other research). They have been tested independently with several sample problems.

One of the subroutines was written in FORTRAN II, at The University of Michigan, as part of another research project. It was adapted for the Berkeley System and then translated (using SIFT) to FORTRAN IV. We call this subroutine LOOK. LOOK is based on ideas presented for the first time in the Westinghouse Scientific Paper 10-1210-1-P1 by R. Hooke and T. A. Jeeves.[5] LOOK was coded in FORTRAN II and presented with several applications in the Westinghouse Scientific Paper 6-41210-1-P1 by C. F. Wood. Our

---

[2] Flood, Merrill M. and Alberto Leon, "A Direct Search Code for the Estimation of Parameters in Stochastic Learning Models," Mental Health Research Institute, The University of Michigan, Preprint 109, May 1963.

[3] Flood, Merrill M. and Alberto Leon, "A Generalized Direct Search Code for Optimization," Mental Health Research Institute, The University of Michigan, Preprint 129, June 1964.

[4] Leon, Alberto, "A Comparison Among Eight Known Optimizing Techniques," Space Sciences Laboratory, Internal Working Paper No. 20, August 1964.

[5] The paper was published subsequently in Journal Assoc. Computing Machinery, 1961, 8, 212-229.

subroutine is a modified version of the original LOOK even though the main ideas remain unchanged. We are indebted to Mr. Wood for a copy of the deck of LOOK's original FORTRAN II code.

The other three subroutines were written in FORTRAN IV for GROPE. These optimizing subroutines are a Simple Random Strategy (RANDOM), a Shrinkage Random Strategy (SHRINK), and the Satterthwaite strategy (SATTER).

Our codes for RANDOM, SHRINK and SATTER are based on ideas presented by D. S. McArthur in the Esso Research and Engineering Report No. RL34M61. We are indebted to Dr. McArthur for a FORTRAN II copy of all his working decks. The fundamental difference between our version and McArthur's original ideas lies in the orientation of our present research. Costs involved in making the experiments and values associated with the same tests are not taken into account here; but they play an important role in McArthur's work. Another important difference is the fact that McArthur's work is mainly stochastic in nature and we are not considering stochastic optimization, at least in this stage of our research. Some additional changes were introduced to make possible the use of these subroutines by GROPE.

There exist three program parameters to guide the optimizing subroutines in some phases of the work which are identical in all of them. These parameters are:

LE(I), a vector of NCODS elements

LUT(I), a vector of NOCDS elements

and LOR

The first executable command of each optimizing subroutine is a computed GO TO statement with two transfer points as follows:

GO TO (n_1, n_2), LOR

LOR is set equal to unity by MAIN and consequently the above statement causes control to be transferred to statement n_1. This statement prints out the name of the incoming subroutine and immediately after that LOR is reset to two suppressing the previous print out from subsequent calls to the same subroutine within the same trial. At the end of a trial, at the beginning of a new problem or before returning from SNICON, whenever it is used, LOR is set equal to unity again. This gives proper identification of the subroutine in use.

It was stated previously that each time the optimizing subroutine in use requires the evaluation of the objective function control must be transferred to OPTIMU which in turn calls EVALT. The vector LE(I) controls the entry point to the subroutine after returning from OPTIMU. LE(I) is set equal to unity at the beginning of the run and also at the end of each trial. Proper value is assigned to element ICODE of the vector LE before leaving for each functional evaluation assuring that return is made to the right point as may be seen in the FORTRAN IV listings.

Some optimizing subroutines establish at the very beginning of the run parameters to be used later on as step sizes, fraction of the operating space to be inspected and the like. However, these parameters are reevaluated during the computation process as a function of earlier results or at definite points during the process. Because of the sequential nature of GROPE one would not want to reuse the initial values of these parameters unless so decided by the subroutine in agreement with the optimizing procedure. In other words, if a subroutine is left and, let us say, certain step size was in use at that time, it is desirable to utilize the same one the next time the subroutine is called back again. The vector IUT(I) exercises the required control to ensure the above condition is satisfied. IUT is set equal to zero by OPTIMU and the element ICODE of IUT reset to unity whenever necessary.

The set of optimizing subroutines actually working with GROPE uses subroutine TESTCO and subroutine TOSS.

Subroutine TESTCO makes it easy to include side conditions present in a problem that are difficult to treat more directly as part of the function to be optimized. TESTCO is called if $KR11 > 0$. The call is executed after changes have been made in the value of the variables by the optimizing subroutine but before calling the evaluating subroutine EVALT. We should make it clear that we distinguish between constraints and side conditions. Upper or lower limits imposed upon the variables are treated here as "constraints," and are controlled by KR6, KR7 and KR9 in the optimizing subroutines. More general restrictions on the variables, represented by equations or inequations such as

$$a \leq x_i + x_j + x_k \leq b, \text{ or}$$

$$c > x_i + x_j > d \ ,$$

are called side conditions and treated within TESTCO.

The above scheme may be unnecessary in new procedures brought into GROPE in the future. They probably have their own way of handling the boundaries problem. Keeping the subroutines as close as possible to the original version eliminates reprogramming and possible errors.

The other subroutine of general use is TOSS. TOSS is called by the optimizing subroutine whenever $KR > 0$, and permutes the components of the vector NORVI randomly. We have found substantial improvements in quality and speed of solution for certain classes of problems when TOSS is used. TOSS uses RAM2, a MAP routine listed in Appendix D. RAM2 is a pseudo-random number generator, for numbers distributed uniformly between zero and unity: RAM2 is borrowed from the Michigan System (in FAP) but the necessary modifications have been made to make RAM2 compatible with the IBJOB monitor at Berkeley, and usable with FORTRAN IV programs.

Brief descriptions of LOOK, RANDOM, SHRINK and SATTER are given below. Flow Charts are included in Appendix C and FORTRAN IV Listings in Appendix D

I. SIMPLE RANDOM (Subroutine RANDOM)

The RANDOM search strategy is presented and discussed by Brooks[6, 7] who suggests this strategy as a helpful one in problems with large number of variables. RANDOM is a good technique to use whenever there is no information on hand about the nature of the problem. It seems inadvisable to use RANDOM with problems involving few variables and expensive tests. The RANDOM strategy is not an iterative method but the mechanics of the code could be considered similar to the ones using iterative procedures.

In SIMPLE RANDOM, tests are simply run at random; that is, the level of each variable is set at random within predetermined limits. No use

[6] Brooks, S. H., "A Discussion of Random Methods for Seeking Maxima," Journal of the Operations Research Society of America, 6, No. 2, 1958.

[7] Brooks, S. H., "A Comparison of Maximum Seeking Methods," Journal of the Operations Research Society of America, 7, No. 4, 1959.

is made of past data in deciding where the test should be run. The highest response attained is accepted as the optimum.

The parameters of RANDOM were explained in the presentation of subroutine RSDTA.

There are two stopping rules in the operation of RANDOM:

a. Whenever the maximum number of cycles without significant improvement is reached;

b. Or, if the above is not reached, at the completion of the total number of cycles.

It is important to say a few words about the meaning of improvement. Improvement is accepted within a cycle so long as it is recognizable in the full significance of the machine. In order for the best point of a cycle to be considered better than the best one of the previous cycle, the point has to be better than the other by a prescribed fraction of it. For example:

If we define SC = best point of present cycle

SP = best point of previous cycle

TOL = fraction of improvement

we say that improvement occurs in a minimization problem, to be specific, if

$$SC \leq SP + TOL * ABSF(SP)$$

Besides the control keys used in the general program SIMPLE RANDOM utilizes KR17 and KR4 to guide calls to subroutine OUTPUT in the way described in Appendix B.

II. <u>SHRINKAGE RANDOM (Subroutine SHRINK)</u>

The Shrinkage Random Method is described and analyzed by Brooks in the references inserted in footnote of page 26.

The shrinkage strategy is, in a sense, the same as SIMPLE RANDOM. Here the searching space is gradually reduced as the research program continues. Tests are run purely at random within the operating limits for NTOS tests. The operating space is then reduced (SHRON) by a fraction, with the best result discovered so far as the center of the remaining operating space. NTOS tests are then run at random within the reduced space. This procedure continues until one of the stopping rules is met.

A list of the parameters used by SHRINK is found together with our description of subroutine RSDTA.

As in SIMPLE RANDOM there exist two stopping rules in SHRINK:

a. Whenever the maximum number of cycles without significant improvement is reached;

b. Or, if the above is not reached, at the completion of the total number of cycles.

The users of the SHRINK idea report that the computer results, in presence of noise, have shown that SHRINK does not show any significant advantage over a purely RANDOM search strategy. It is our experience, however, that in problems where noise is absent and the objective function is not ill-behaved, SHRINK performs better than SIMPLE RANDOM. These results are in agreement with what we had expected.

The rules for improvement are the same as in RANDOM. The options for output printing are the same also and are controlled by the same keys.

III. SATTERTHWAITE STRATEGY (Subroutine SATTER)

The Satterthwaite Strategy is fully described by Satterthwaite[8] and

[8]Satterthwaite, F. E., "REVOP or Random Evolutionary Operation," Statistical Engineering Institute, Report No. 10/10/59.

is just one of the many possible ways of introducing the random search ideas in optimization strategies.

The strategy works as follows: "A starting point is picked at random* and a test is run. A direction is then also picked at random (in the N-dimensional space) and an experiment is run one step in that direction. If this gives a result which is poorer than the first result, a step is taken in the opposite direction and another test is run. If this result is better than the original experiment, additional steps are taken in the same direction until an optimum has been exceeded. Another direction is then picked at random, starting from this optimum point, and steps are taken along the new vector until another maximum has been exceeded" (Esso Research Report).

As McArthur points out, this strategy differs from the other strategies in the way things are handled at the boundaries. Satterthwaite suggests that if a vector reaches a boundary it should be rebound from the boundary rather than stopping at it. There are many other ways of solving the situation at the boundaries depending upon the characteristics of the problem as well as on the nature of the constraints. BEST UNIVAR, as an example, uses a compromise between stopping at the boundary and returning inside the region.

The parameters required by SATTER are listed in the description of subroutine RSDTA. As in the previous strategies there are two stopping rules in SATTER:

a. Whenever the maximum number of changes in direction without significant improvement is reached;

*Or at the discretion of the experimenter.

b. Or, if the previous one is not met, at the completion of the total number of experiments.

SATTER accepts improvement, going in the same direction, so long as it is recognizable in the full significance of the machine. Following the same ideas of RANDOM and SHRINK, in order for the best point of a particular direction to be considered better than the best point of the previous direction, the point has to be better than the other by a prescribed fraction of it.

IV. PATTERN SEARCH STRATEGY (Subroutine LOOK)

LOOK is fully described in the Westinghouse publications mentioned earlier in this paper.

LOOK may be described briefly as follows:[9]

1. Initialization--A starting point for the search is calculated[*] and stored.

2. Exploratory Search--Various moves are made to determine a desirable direction for the search. Any move which is better than the reference value is kept and becomes the new reference value. On the initial entry or whenever the exploratory search is not immediately preceded by a pattern move, the reference value is the last base point. Following a pattern move, the reference is the value at the end of the pattern move.

3. Success?--If the best value found for the function during the exploratory search is better than its value at the last base point, a new base point is established. Otherwise, the last base point is restored.

---

[9]Wood, C. F., "Recent Developments in 'Direct Search' Techniques," Westinghouse Research Report 62-159-522-R1.

[*]Or given as data.

4. Save base point and make Pattern Move--The latest functional value replaces the previous value and the corresponding values of the independent variables do likewise. This establishes a new base point. The pattern move is generated by moving each independent variable away from the la est base point value by an amount equal to the difference between the old and new base point values. A pattern move is always followed immediately by an exploratory search.

5. Restore Last Base Point--The independent variables are set at the values corresponding to the last base point. The functional value for the same point becomes the initial reference for testing the individual moves of the exploratory search.

6. Had Pattern move just been made--If the exploratory search preceding the failure was itself preceded by a pattern move, perform another exploratory search. Otherwise, check for search completion.

7. Can step size be reduced--If the step sizes for all the independent variables are at their minima, the search is complete. Otherwise, reduce step size and perform another exploratory search.

Our version of LOOK presents, as was said before, some differences with the above description.

One of the modifications concerns the order in which the variables are to be analyzed. The original LOOK examines the variables always in the same regular order, while in our version the experimenter has control over the order by means of a vector read in as data.

Another major change gives access for controlling the generation of the pattern move. In the original LOOK the pattern vector is equal to the difference between the old and new base point values. Our version multiplies this difference by a factor (DELTA). Moreover, if the pattern move is successful DELTA is increased by SHRUN the next time.

An option to inspect just a portion of the set of variables is also available in our version together with the possibility of making tests of grouping effects by performing the optimization with different subsets of the entire set of variables.

The parameters which are necessary for the operation of LOOK are presented in our description of subroutine RSDTA.

The final termination of the search is made when the step size is sufficiently small (DELMIN) to insure that the optimum has been closely approximated. In any case, the step size must be kept above a practical limit imposed by the means of computation. The search is stopped when two conditions occur at the same time, namely (1) the step size is at minimum (DELMIN) and (2) the forward and reverse moves of all independent variables fail following a base point test failure.

As in the techniques described previously there are two criteria for move evaluation:

a. Evaluation of individual moves: here, an improvement is accepted so long as it is recognizable in the full significance of the machine;

b. Test to determine whether a new base point has been found: in the base point test, the point has to be better than the last by a prescribed fraction of the last point. As before, this fraction is given by TOL. As Hooke and Jeeves say, "In practice, pattern search has proved particularly successful in locating minima on hypersurfaces which contain 'sharp valleys'. On such surfaces classical techniques behave badly and can only be induced to approach the minimum slowly."

Internal options are available for testing boundaries and output printing. Control keys exercise the control of these options.

OPERATION OF GROPE

GROPE uses to attain its second objective, at least in its present stage, the Overlay Feature of IBLDR (The Loader Operating under the IBJOB monitor, Berkeley System).

"Overlay[10] is a method of core storage utilization by jobs that exceed the capacity of core storage. The programmer divides the job to be executed by the overlay method into links. A link is one or more decks* of a job. One of these links, called the main link, is loaded directly into core storage and remains in core storage throughout the execution of a job along with the Overlay subroutine and the tables required for execution. The Loader writes the other links, called dependent links, on some external file. A dependent link is one that is usually in core storage only at the time it is being used. It can be overlaid by other dependent links."

We include in Appendix E three possible overlaying schemes of GROPE. Some of the advantages of each one of them will be pointed out later on in this section together with their operating differences. In the figures of Appendix E, the vertical lines represent a link, or links, into which the program has been divided. Each link may contain one or more decks or subroutines. The horizontal lines indicate the logical origin of the links. Link 0 is the main link, and remains in high-speed core storage at all times; the other links are stored in some external file.

---

[10]Users Manual, University of California, Computer Center, Berkeley, Chapter VII, Section J (The Loader).

*Programs or subprograms.

"Overlay[11] can be induced only by the execution of a CALL from a link that is presently in core storage to a link that is not in core storage. When a CALL statement is executed in a link to any of the decks contained in another link, the incoming link replaces the link in core storage that has the same logical origin as the incoming link, and will also overlay all deeper links in the same chain below that logical origin. A chain is a sequence of links in core storage from the deepest link required, through whatever links precede it, to the main link. It is assumed that the normal way of terminating the execution of a deck in a dependent link will be with the RETURN statement."

Referring to Scheme A of Appendix E, as an example, the possible configuration of links in core at any given time is:

(1) Link 0 (main link) only

(2) Link 0 and Link 1

(3) Link 0 and Link 2

(4) Link 0, Link 2 and Link 3

(5) Link 0, Link 2 and Link 4

(6) Link 0, Link 2, Link 4 and Link 5

(7) Link 0, Link 2, Link 4 and Link 6

(8) Link 0, Link 2, Link 4 and Link 7

(9) Link 0, Link 2, Link 4 and Link 8

Figure No. 4 of Appendix E shows a schematic representation of the above overlay structure as it is assigned to high-speed core storage. The input/output buffers will occupy the unused high-speed core storage area

[11]See 10, p. 36.

between the longest possible link configuration, Number 8 of the example presented, and the highest available core storage location. The FORTRAN COMMON area, if used, will be assigned following the library subroutines.

Three Different Overlaying Structures

We have been using the overlaying feature as an instructive and necessary exercise even though the capacity of high-speed core memory has not yet been exceeded. However, it is expected we will face this problem very soon with the addition of new optimizing subroutines and alternate adaptive mechanisms.

Many are the overlaying schemes GROPE can use; three of the many possible ones are illustrated in Appendix E.

The presence of a high number of links provides space in high-speed core memory either for bigger or more optimizing subroutines. However, the existence of too many links requires a lot of loading and unloading operations in the high-speed core memory unit. These operations increase considerably the execution time. We include next, as an illustrative example, some of the operational characteristics presented by the group of structures of Appendix E during the solution of a particular problem.

a. No overlaying feature:

Execution time: 185.2 seconds

Number of trials: 37

Functional Evaluations: 7035

b. Scheme "C":

Execution time: 190.7 seconds

Link 1: loaded 10 times, unloaded 10 times

Link 2: loaded 8 times, unloaded 8 times

Link 3: loaded 7 times, unloaded 6 times

Link 4: loaded 8 times, unloaded 8 times

Number of trials: 37

Functional Evaluations: 7035

c. Scheme "B":

Execution time: 403.8 seconds

Link 1: loaded 2 times, unloaded 2 times

Link 2: loaded 2 times, unloaded 1 time

Link 3: loaded 4 times, unloaded 4 times

Link 4: loaded 1 time, unloaded 1 time

Link 5: loaded 1 time, unloaded 1 time

Link 6: loaded 3 times, unloaded 3 times

Number of trials: 13

Functional Evaluations: 2693

Problem unfinished (Actual execution time exceeded allowed time for execution.)

d. Scheme "A":

Execution time: 385.2 seconds

Link 1: loaded 1 time, unloaded 1 time

Link 2: loaded 1 time

Link 3: loaded 2 times, unloaded 2 times

Link 4: loaded 2 times, unloaded 2 times

Link 5: loaded 1 time, unloaded 1 time

Link 6: loaded 1 time

Number of trials: 2

Functional Evaluations: 457

Problem unfinished (Actual execution time exceeded allowed time for execution.)

The following figures illustrate the increases in available space for optimizing subroutines with each one of the previous structures. It must be said that the available space in high-speed core memory is also a function of the size of subroutine EVALT. The size of EVALT in the specific case we are describing is 13,361 octal locations (up to 100_8 locations must be reserved for input/output buffers).

Scheme	Highest used core location	Highest usable core location
No overlaying	61410_8	77013_8
C	55147_8	77013_8
B	52443_8	77013_8
A	51736_8	77013_8

The highest core storage location used by the program without the optimizing subroutines (i.e., Link 0 plus the system routines in Scheme "C") is 53156_8. The biggest of the optimizing subroutines (LOOK) requires 2021 octal locations. We have then an approximate number of 22000_8 locations available for optimizing subroutines whenever using a subroutine EVALT of 13361_8 locations. This means that the biggest optimizing subroutine we can load into high-speed memory using Scheme "C" is one of 22000_8 size. If we come across a bigger routine it will be necessary for us to choose a different overlaying structure.

When we talk about the size of subroutine EVALT we imply the subroutine itself plus the associated ones it may require for its proper work.

FLEXIBILITY AND LIMITATIONS

GROPE as it is written now is adequate to treat difficult optimization problems involving as many as 100 variables.

As was previously stated the program has some subroutines that the user must write and compile in order to treat his particular optimization problems. These subroutines are EVALT to evaluate the function being optimized, the additional routines required by EVALT, and TESTCO whenever side conditions are treated in the way described in our presentation of subroutine TESTCO. GROPE is written as a minimization code, arbitrarily and without loss of generality since minimizing a function F is the same as maximizing -F. Nevertheless, the user must have this fact in mind since the tests for improvement in the functional value follow the minimization criterion.

The following subroutines should be modified if additional optimizing subroutines are added to GROPE:

1. RSDTA, including the parameters required by the incoming subroutine or subroutines.
2. OPTIMU providing the calling statements for the new subroutine or subroutines. CLOCK readings and counting vectors must be included also.

3. OUTPUT with the desired variables and parameters to be printed out in accordance with our description of subroutine OUTPUT.

4. The new optimizing subroutine or subroutines.

It must be recalled that the actual maximum possible number of optimizing subroutines is 20. If more than 20 subroutines are to be present further changes in the dimension statements (LE, KEY, LIZ, NT, ISH, LOCK, PROB, SPROB, RNB, and LUT) are required.

To incorporate additional adaptive mechanisms, proper changes must be introduced in subroutine RCPRO and subroutine LPRMT. Additions to subroutine OUTPUT may be desirable in these cases.

New decision criteria for switching from one optimizing subroutine to another will ask for modifications in subroutines IDECI, DECI and OPTIMU. Since IDECI and DECI handle the initialization of the chosen criterion and subsequent tests respectively, they are supposed to include the incoming criteria. Subroutine OPTIMU carries out counting operations and the like so it probably requires some changes as well. No changes are necessary in OPTIMU provided the new decision criteria use in some way the number of functional evaluations or the amount of time at work. These criteria are already incorporated in the program by means of variables NEL and NEL1 and vectors ISH and NT.

Core storage aspects were described briefly in the section entitled "Operation of GROPE".

It is understood that changes in the program must come together with proper modifications in the FORTRAN COMMON statement. It is advisable to follow very close the instructions concerning "The COMMON Statement" given in the IBM reference manuals whenever changes in the COMMON blocks are to be introduced. The variables were assigned to each block (with some symbolic name) trying to reduce as much as possible the number of future changes. Hence all the parameters associated with a particular subroutine, for instance, are in one COMMON block (e.g., COMMON/BBP/ for subroutine LOOK).

REMARK

The program as described in this paper is completely debugged and has been tested with several sample problems. However, we have not had enough experience as yet to permit us to make conclusive statements about its work. A large number of different users with different areas of interest are necessary now to improve the program and to meet the goals proposed in our Introduction.

The code has been presented under the assumption that algebraic expressions for the optimization function, the side conditions, and the constraints are all available. This was for purposes of exposition only, for all that is really necessary is to have subroutine EVALT--at least with the present group of optimizing subroutines. For example, EVALT might be a simulation of some process that would yield functional values but without having the function available in algebraic form at all. This aspect deserves careful attention while deciding about additional optimizing subroutines to be brought into GROPE. Most of the known optimizing procedures require algebraic expressions for the objective function except the so-called direct search techniques (LOOK, BEST UNIVAR, RANDOM, etc.). The need for such algebraic expressions might very well restrict the actual scope of GROPE, its generality and its flexibility.

There are several important aspects of GROPE deserving further research. The choice of decision criteria for switching optimizing subroutines is a crucial one. Also of great importance is the use of accelerating convergence procedures both as components of the optimizing subroutines and as independent devices to guide and link the work of successive subroutines. A more flexible feature than the overlaying structures will become necessary with the addition of other optimizing techniques. These are only a few among the problems we face in our future research on GROPE.

APPENDIX A

Glossary of Parameters, Variables and Subroutine Names[*]

A1 (RCPRO),	Structure parameter of the Adaptive Mechanism.
A2 (RCPRO),	Structure Parameter of the Adaptive Mechanism.
ADAPT,	Name of the subroutine handling the Adaptive Mechanism of the program.
B1 (RCPRO),	Structure parameter of the Adaptive Mechanism.
B2 (RCPRO),	Structure parameter of the Adaptive Mechanism.
BESV (ADAPT, OPTIMU),	Vector yielding the best functional value of present trial.
C1 (RCPRO),	Structure parameter of the Adaptive Mechanism.
C2 (RCPRO),	Structure parameter of the Adaptive Mechanism.
DAMBL (SNICON),	Relaxation parameter in linear and non-linear programming problems.
DECI,	Name of subroutine deciding whether or not the end of trial has been reached.
DEL (LOOK),	Quantity specifying incremental change in all variables.
DELMIN (LOOK),	Minimum allowable value of DEL.
DELR (LOOK),	Ratio by which DEL is to be changed.
DELTA (LOOK),	Quantity specifying size of pattern move.
EP1 (ADAPT),	Parameter to define when a functional value is better than the previous one.
EVALT,	Name of subroutine yielding functional values.

[*]The names in parentheses stand for the subroutine or subroutines using the variables.

FMAX (ADAPT, OPTIMU),	Best functional value of present trial.
I1 (ADAPT),	Number of times the optimizing subroutines have been unsuccessful in sequence.
IBEST (ADAPT),	Number of the trial at which the optimum point was reached.
IC1 (OPTIMU),	Variable to store clock readings temporarily.
IC2 (OPTIMU),	Variable to store clock readings temporarily.
ICODE (ADAPT, OPTIMU),	Number of the optimizing subroutine in use at any given time.
IDECI,	Name of subroutine initializing the switching decision criteria.
IN (ADAPT),	Three-integer element vector to initialize the pseudo-random number generator.
IN1 (TOSS),	Three-integer element vector to initialize the pseudo-random number generator.
IN2,	Three-integer element vector to initialize the pseudo-random number generator. Available to the user.
IN3,	Three-integer element vector to initialize the pseudo-random number generator. Available to the user.
IROUT (ADAPT),	Number of the optimizing subroutine in use during trial IBEST.
ISH (OPTIMU),	Counting parameter denoting the number of calls to each optimizing subroutine.
ITRIA (ADAPT),	Number of the present trial.
ITIME (IDECI, DECI),	Maximum allowed time per trial in each optimizing subroutine.
J1 (ADAPT),	Counting parameter indicating the number of the optimizing subroutine selected for next use.
JRES (ADAPT),	Vector with the number of the subroutine used in each one of the trials.

JSTI (ADAPT),	Vector of "stimulus".
KEY (ADAPT, OPTIMU),	Vector denoting the condition causing return from each one of the optimizing subroutines.
K6, K7,	Logical Tape Numbers.
LE (OPTIMIZING SUB.),	Controls the entry point to the subroutine after returning from OPTIMU.
LGO (OPTIMU),	Controls the entry point after returning from DECIS.
LIZ (ADAPT),	Parameter controlling calls to optimizing subroutine after unsuccessful use.
LL1 (RCPRO),	Controls the entry point to the subroutine.
LL2 (EVALT),	Controls the entry point to the subroutine.
LL3 (OUTPUT),	Controls the entry point to the subroutine.
LL4 (OPTIMU),	Controls the entry point to the subroutine.
LL5 (SNICON),	Controls the entry point to the subroutine.
LL6 (ADAPT),	Controls the entry point to the subroutine.
LMDS (RCPRO),	Identifies the adaptive mechanism used whenever several are available.
LOCK (ADAPT),	Parameter controlling calls to optimizing subroutines after unsuccessful use.
LOOK,	Name of optimizing subroutine using pattern search strategy.
LOR (OPTIMIZING SUB.)	Controls the entry point to the subroutine.
LPRMT,	Name of subroutine to read the parameters of the adaptive mechanism in use.
LT (MAIN),	Controls the entry point after coming from SNICON.
LT11 (MAIN),	Stores clock readings temporarily.
LT22 (MAIN),	Stores clock readings temporarily.
LT33 (MAIN),	Stores clock readings temporarily.

IUT (OPTIMIZING SUB.),	Exercises the required control to ensure that the same program parameters are used in future calls to the optimizing subroutine, whenever so desired.
M (MAIN),	Number of equations or inequations in linear problems.
MCYCLE (RANDOM),	Maximum number of cycles the subroutine can be used.
MCYCLO (SHRINK),	Maximum number of cycles to be performed.
MRDTN (SATTER),	Maximum number of cycles to be performed.
MXNEL (IDECI, DECI),	Maximum number of functional evaluations allowed within a trial.
MXTIM (DECI, IDECI),	Maximum amount of time allowed for every one trial.
MXTRI (ADAPT),	Maximum number of trials to be performed.
N (MAIN),	Number of variables.
NC (BATCHE),	Not included in the program yet.
NCODS (ADAPT, RCPRO),	Number of optimizing subroutines available.
NDECI (IDECI, DECI),	Identifies the criterion for switching subroutines, whenever several are available.
NDTN (SATTER),	Maximum number of continuous changes in direction accepted without significant improvement.
NEL (OPTIMU),	Number of functional evaluations within a trial.
NEL1 (OPTIMU),	Total number of functional evaluations.
NGO (OPTIMU),	Controls the entry point to the subroutine.
NLOOK (OPTIMU),	Total number of calls to optimizing subroutines.

NT (OPTIMU),	Total time spent by each optimizing subroutine in each independent problem.
NTC (OPTIMIZING SUB.),	Maximum number of variables to be analyzed.
NTES (RANDOM),	Maximum number of functional evaluations within a cycle.
NTI10 (MAIN),	Stores clock readings temporarily.
NT1S (SATTER),	Total number of functional evaluations to be performed in a cycle.
NTIME (MAIN),	Stores clock readings temporarily.
NTNCOS (SHRINK),	Maximum number of continuous cycles accepted without significant improvement.
NTNCYS (RANDOM),	Maximum number of continuous cycles accepted without significant improvement.
NTSNC (MAIN),	Maximum number of times subroutine SNICON can be used.
NTOS (SHRINK),	Maximum number of functional evaluations per cycle.
NORVI (OPTIMIZING SUB.),	Order in which the variables will be analyzed.
OMEGA (SNICON),	An exponential reduction factor taking SHRUN to Ω SHRUN when SNICON is called more than once.
OPTIMU,	Name of the subroutine calling the different optimizing subroutines.
OUTPUT,	Name of the subroutine to print whenever called most of the information available at the calling time.
PBESV (ADAPT, OPTIMU),	Vector yielding best functional value so far.
PFMAX (ADAPT, OPTIMU),	Best functional value so far.
PN (OPTIMU),	Starting vector.

PNMAX (OPTIMIZING SUB.),	Upper limit for each variable.
PNMIN (OPTIMIZING SUB.),	Lower limit for each variable.
PROB (RCPRO),	Present state probabilities of the system.
RANDOM,	Name of the optimizing subroutine using the simple random strategy.
RCPRO,	Name of the subroutine to recompute the state probabilities at the end of each trial.
RFMTS,	Name of the subroutine to read variable formats.
RISTK (SNICON),	Maximum allowable error in satisfying the linear inequalities in mathematical programming problems.
RNB (ADAPT),	Vector of random numbers.
RO (SNICON),	Adjusting factor for TAU in mathematical programming problems.
RSDTA,	Name of the subroutine to read input data.
RWSUB,	Name of the subroutine to read and print out the names of the subroutines used in the program.
SATTER,	Name of the optimizing subroutine using Satterthwaite strategy.
SC(OPTIMIZING SUB.),	Stores best functional value of present cycle temporarily.
SHRINK,	Name of the optimizing subroutine using the shrinkage random strategy.
SHRON (SHRINK),	Fraction by which the operating space is reduced after every complete cycle.
SHRUN (LOOK),	Adjusting factor for DELTA.
SN (OPTIMU, EVALT),	Functional value at any given time.

SNICON,	Name of the subroutine which provides means for repeating an optimization calculation with altered parameter values.
SP (OPTIMIZING SUB.),	Stores best functional value of previous cycle temporarily.
STEP (SATTER),	Step size.
TAU (SNICON),	Adjusting factor for the relaxation parameter DAMBL, calculated for mathematical programming problems as a function of the actual error and the maximum permissible error.
TEMPE (MAIN),	Vector of residuals in linear problems.
THETA (SNICON),	An exponential reduction factor taking DELMIN to THETA*DELMIN when SNICON is called more than once in a linear problem.
TOL (OPTIMIZING SUB.),	Fraction of acceptance for improvement in functional value between two consecutive cycles of one optimizing subroutine.
TOLS (MAIN),	Smallest error accepted in the solution of systems of linear equations.
TOSS,	Name of the subroutine to permute the components of the vector NORVI randomly.

APPENDIX B

Use of Control Parameters*

KR (LOOK)

$KR \leq 0$ Do not call TOSS

$KR > 0$ Call TOSS

KR1 (EVALT)

$KR1 \leq 0$ Do not write out the basic information of the problem on hand

$KR1 > 0$ Write out the previous information

KR2 To be used at the discretion of the user

KR3 To be used at the discretion of the user

KR4 (OPTIMIZING CODES)

$KR4 > 0$ Call OUTPUT at the end of each cycle

$KR4 \leq 0$ Do not call OUTPUT

KR5 (ADAPT)

$KR5 < 0$ Call OUTPUT at the end of each trial and print out the program parameters.

$KR5 = 0$ Call OUTPUT at the end of each trial suppressing the print out of the program parameters

$KR5 > 0$ Do not call OUTPUT

KR6 (OPTIMIZING CODES)

$KR6 \leq 0$ Do not test for limits on variables

$KR6 > 0$ Test for limits

*The names in parenthesis stand for the subroutines using the control parameter.

KR7 (OPTIMIZING CODES)

KR7 $\leq$ 0 Do not test for upper limits

KR7 > 0 Test for upper limits

KR8 (MAIN)

KR8 Number of problems to be solved in a single computer run

KR9 (OPTIMIZING CODES)

KR9 $\leq$ 0 Do not test for lower limits

KR9 > 0 Test for lower limits

KR10 (OPTIMIZING CODES)

KR10 < 0 Call OUTPUT after one internal stopping rule has been reached and print program parameters

KR10 = 0 Call Output after stopping rule has reached suppressing the print out of program parameters

KR10 > 0 Do not call OUTPUT

KR11 (OPTIMIZING CODES)

KR11 $\leq$ 0 Do not call TESTCO

KR11 > 0 Call TESTCO

KR12 (OPTIMU)

KR12 $\leq$ 0 Do not call OUTPUT at the beginning of OPTIMU

KR12 > 0 Call OUTPUT

KR13 (MAIN)

KR13 $\leq$ 0 Do not print out the residuals in linear problems

KR13 > 0 Print out the residuals

KR14 (RSDTA)

KR14 $\leq$ 0 Print out the program parameters read in as data.

KR14 > 0 Do not print out the program parameters

KR15 (RSDTA)

$KR15 \leq 0$ Do not call RFMTS

$KR15 > 0$ Call RFMTS

KR16 (MAIN)

$KR16 > 0$ Call SNICON

$KR16 \leq 0$ Do not call SNICON

KR17 (OPTIMIZING CODES)

$KR17 > 0$ Call OUTPUT after each functional evaluation

$KR17 \leq 0$ Do not call OUTPUT

KR18 (OUTPUT)

$KR18 \leq 0$ Print out actual vector and functional value together with the best previous ones

$KR18 > 0$ Do not print out the best previous values

KR19 (MAIN)

$KR19 \leq 0$ Do not print out the vectors of responses and stimuli

$KR19 > 0$ Print out the above information

KR20 (OPTIMU)

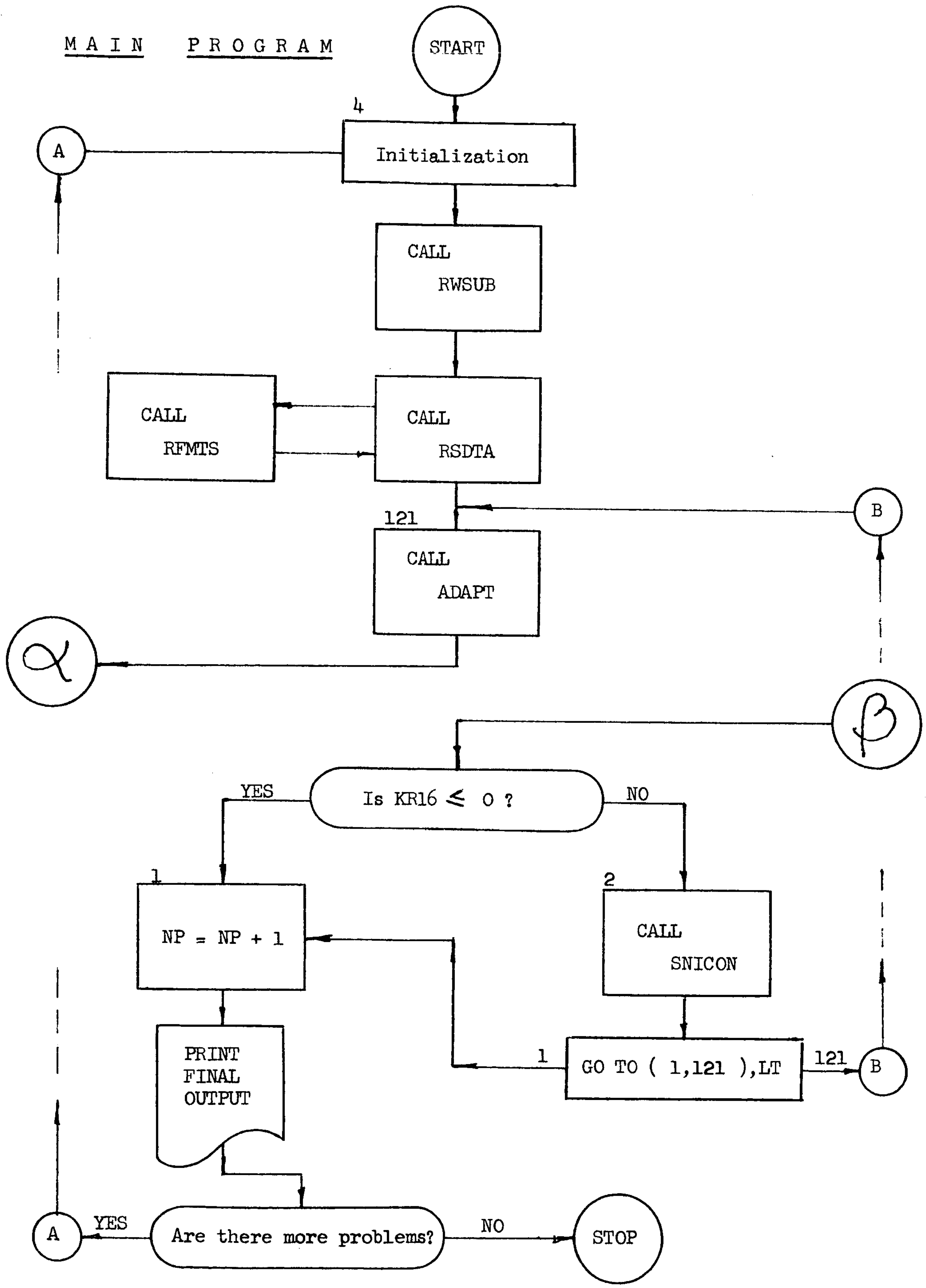
$KR\ 20 < 0$ Call OUTPUT at the end of OPTIMU whenever transfer is caused by internal stopping rule of optimizing code and print out program parameters

$KR20 = 0$ Call OUTPUT suppressing the print out of program parameters

$KR20 > 0$ Do not call OUTPUT at the end of OPTIMU

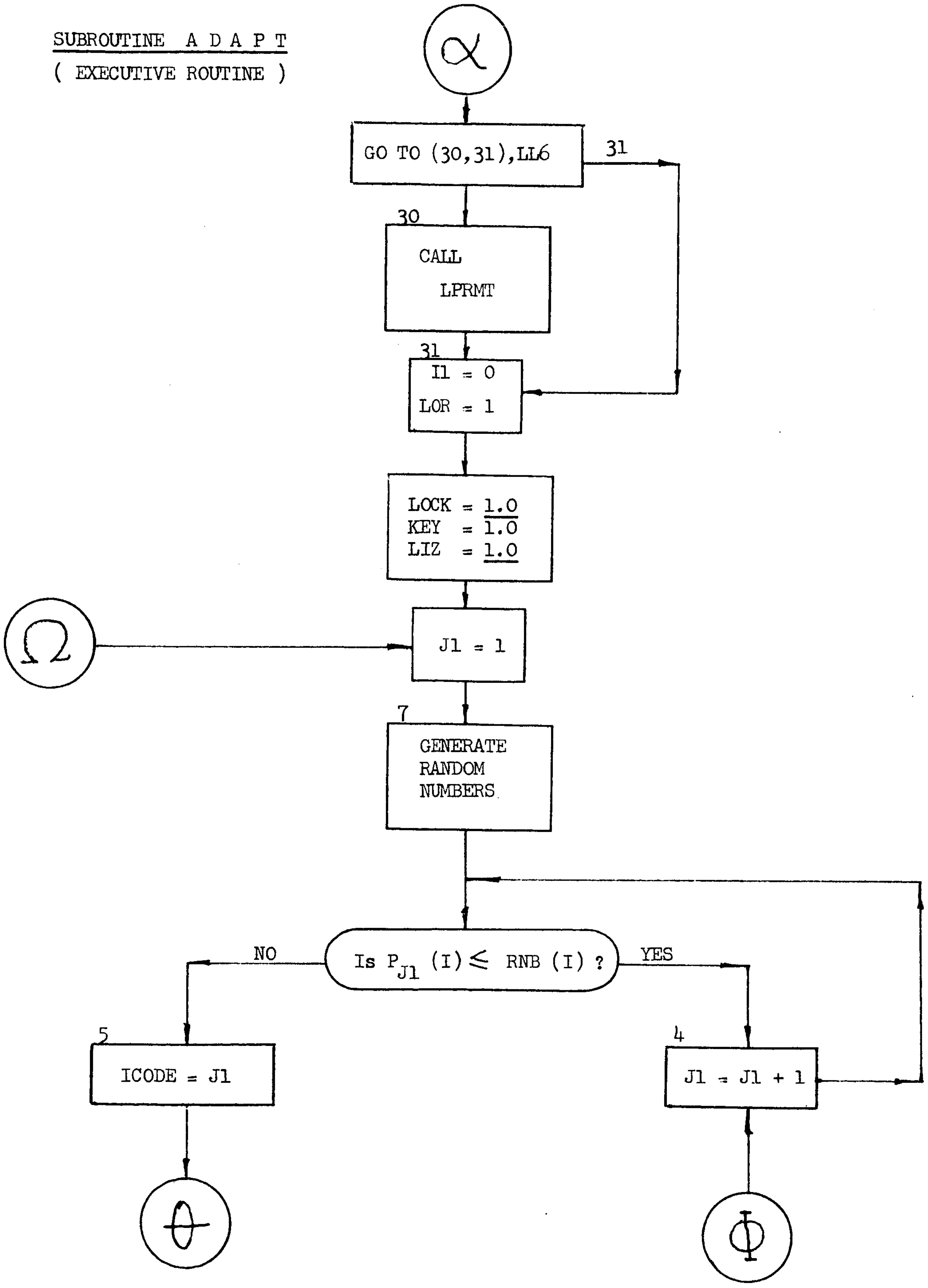
APPENDIX C

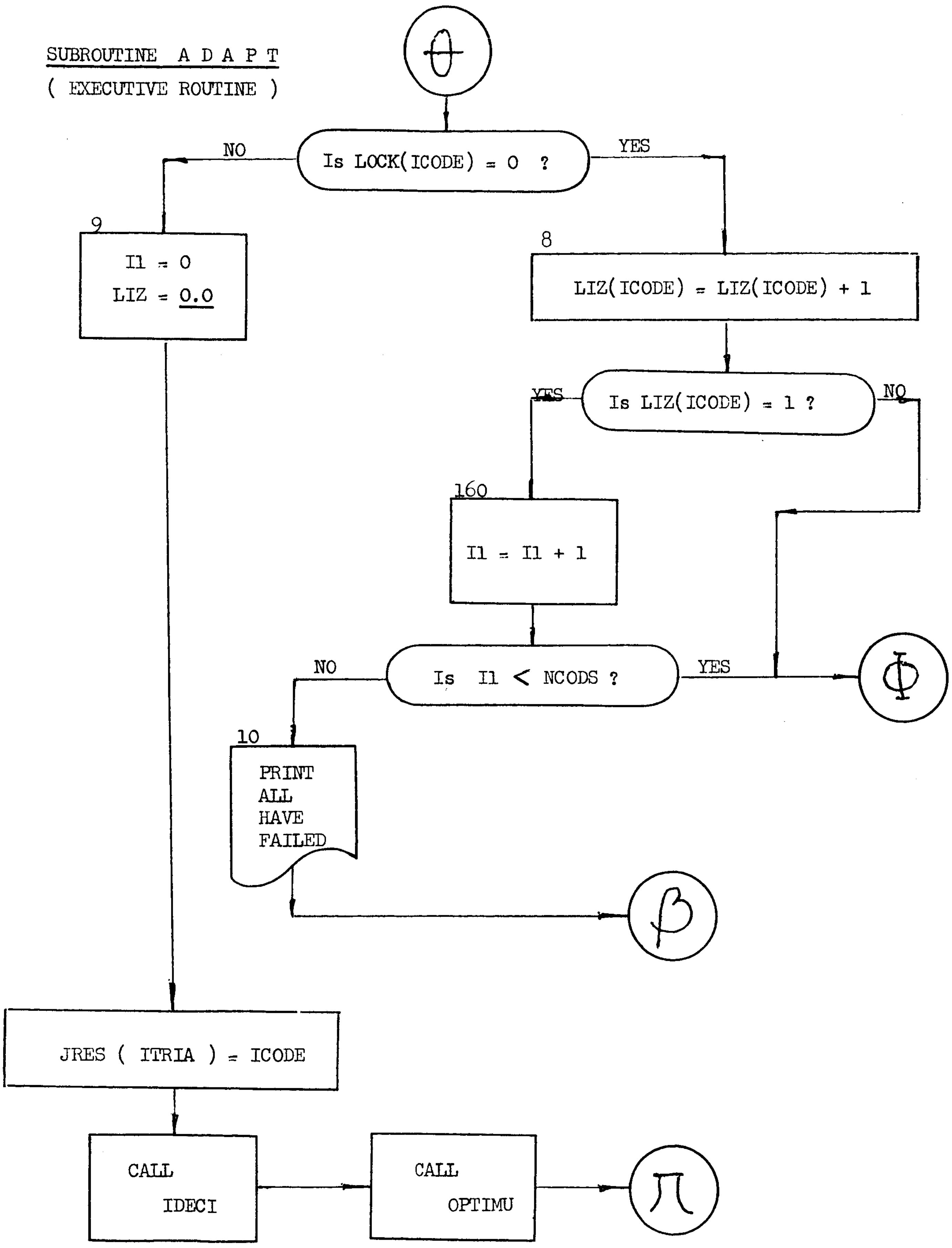
FLOW CHARTS


MAIN PROGRAM
START
4
Initialization
A
CALL RWSUB
CALL RFMTS
CALL RSDTA
121
CALL ADAPT
B
α
β
Is KR16 ≤ 0 ?
YES
NO
1
NP = NP + 1
2
CALL SNICON
PRINT FINAL OUTPUT
1
GO TO (1,121),LT
121
B
A
YES
Are there more problems?
NO
STOP

SUBROUTINE A D A P T

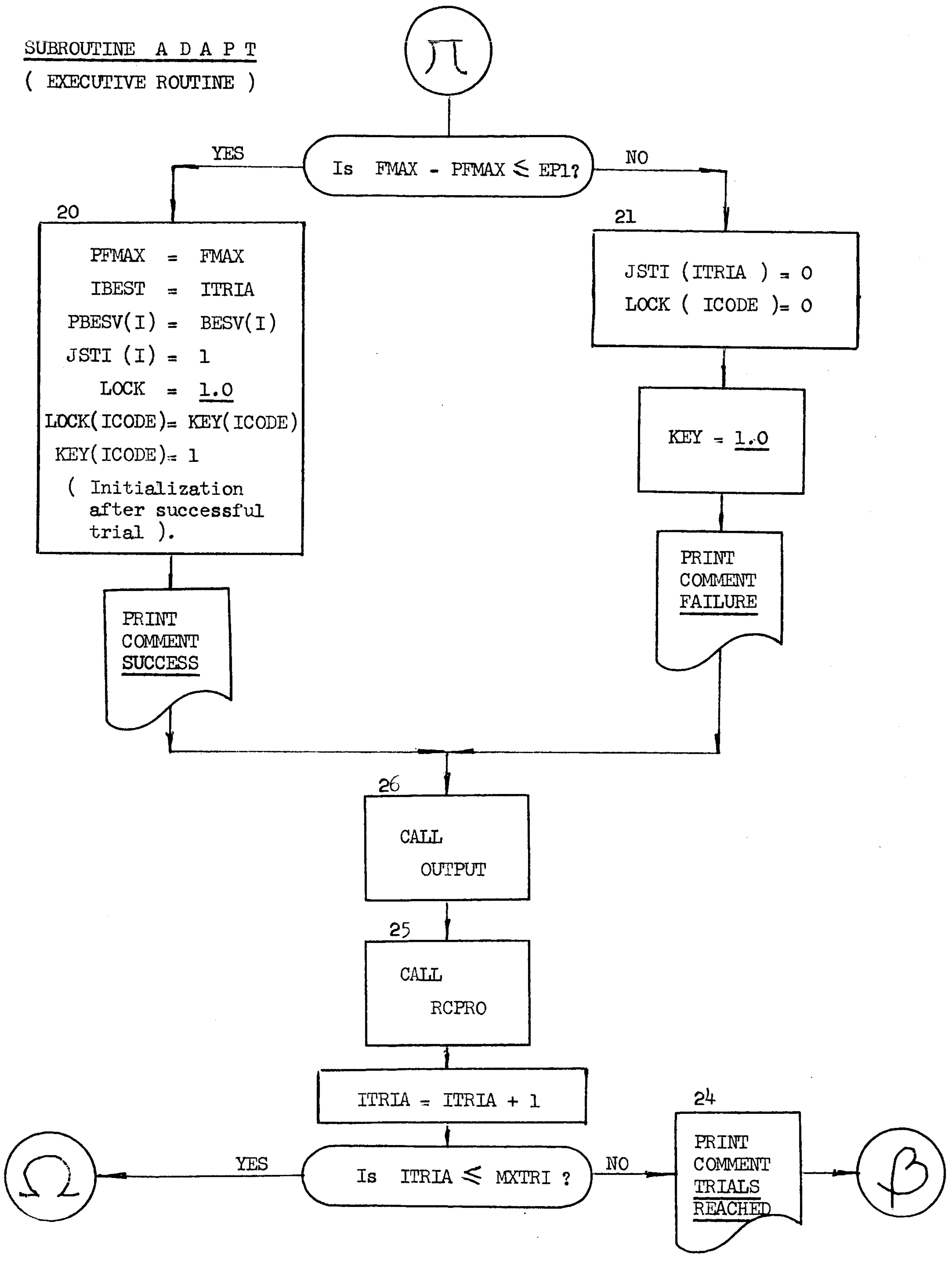
(EXECUTIVE ROUTINE)



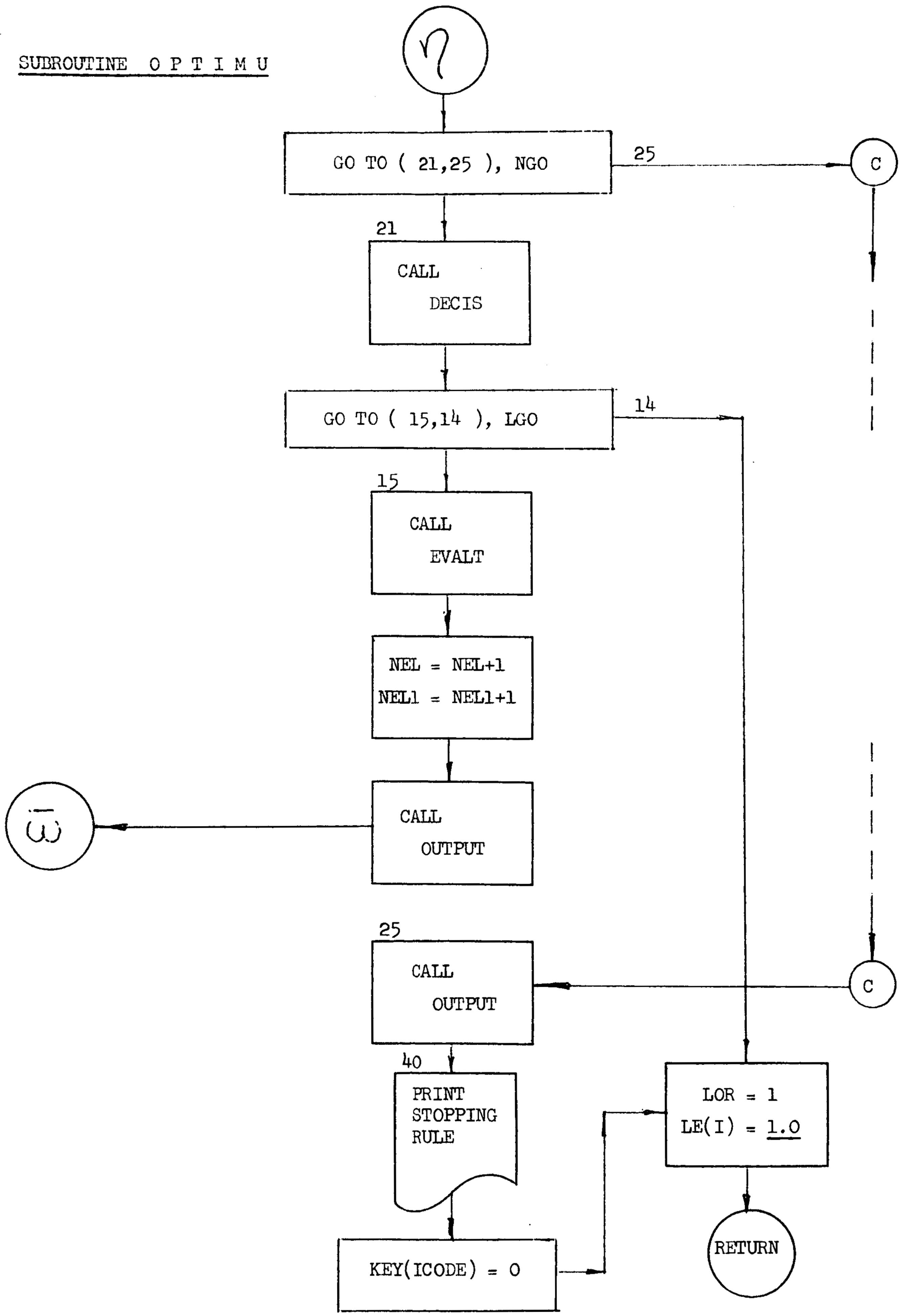

SUBROUTINE A D A P T
(EXECUTIVE ROUTINE)
θ
Is LOCK(ICODE) = 0 ?
NO
YES
9
I1 = 0
LIZ = 0.0
8
LIZ(ICODE) = LIZ(ICODE) + 1
Is LIZ(ICODE) = 1 ?
YES
NO
160
I1 = I1 + 1
Is I1 < NCODS ?
NO
YES
Φ
10
PRINT ALL HAVE FAILED
β
JRES (ITRIA) = ICODE
CALL IDECI
CALL OPTIMU
π

SUBROUTINE A D A P T

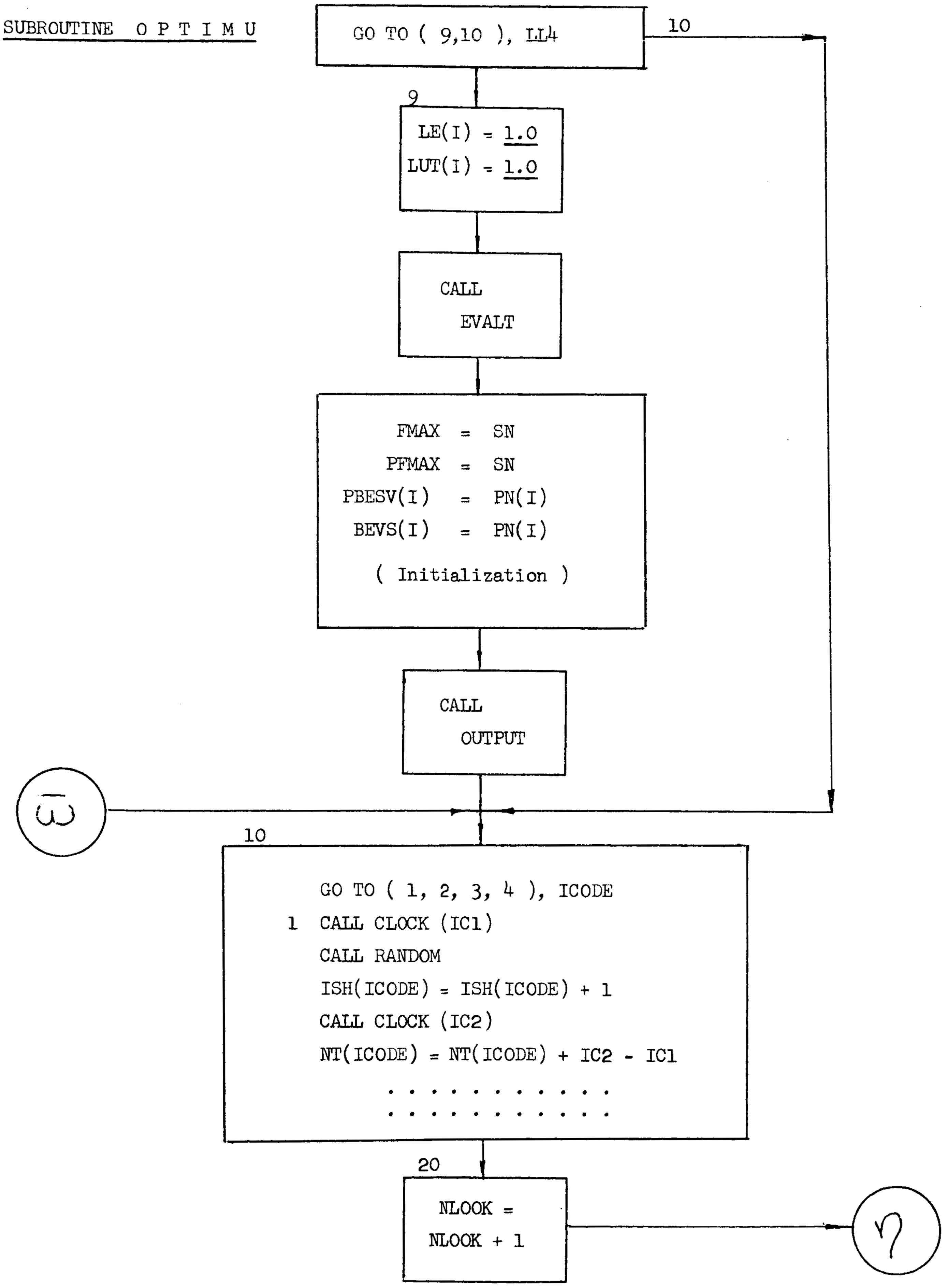
(EXECUTIVE ROUTINE)



SUBROUTINE O P T I M U

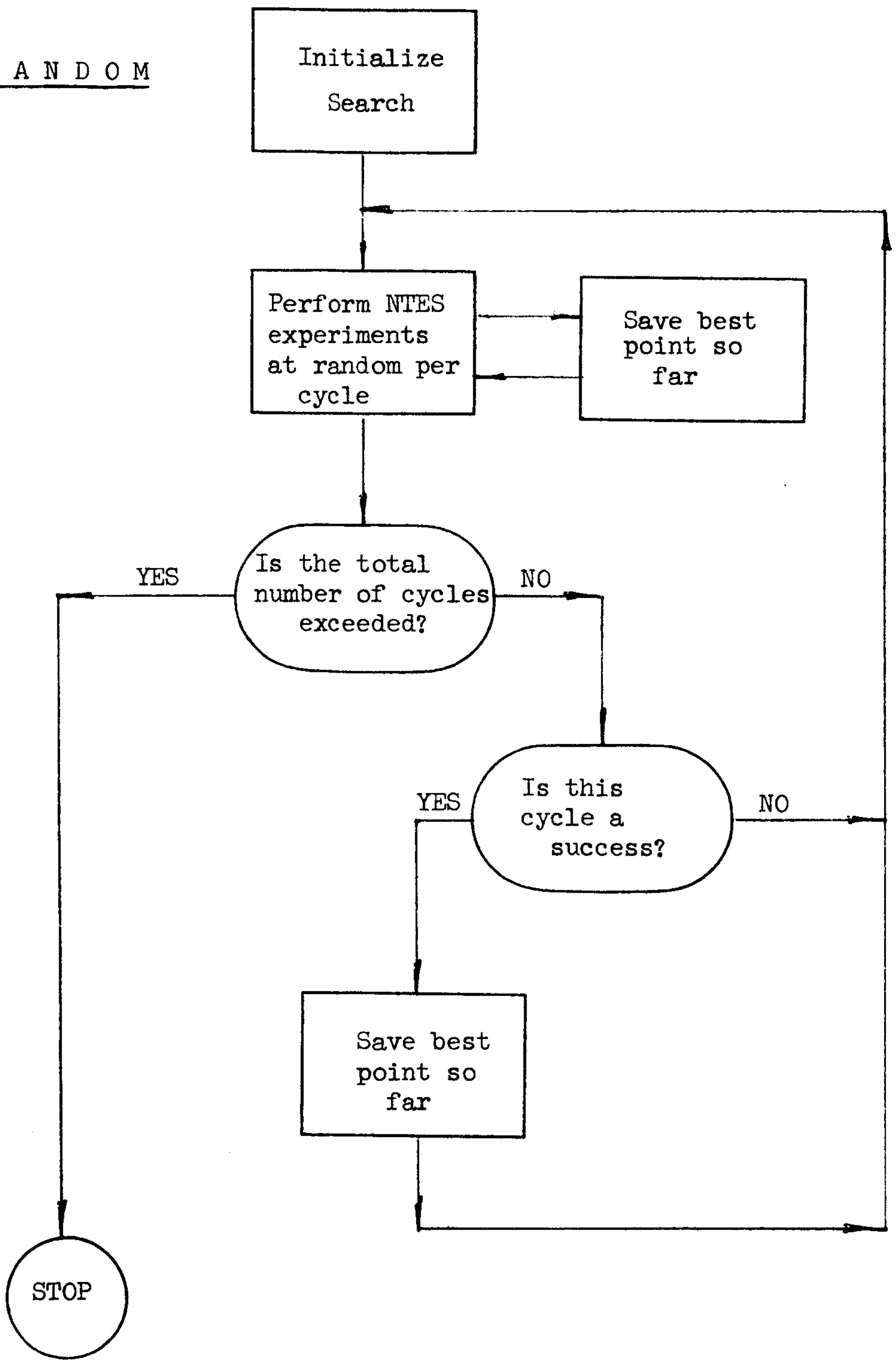


SUBROUTINE O P T I M U

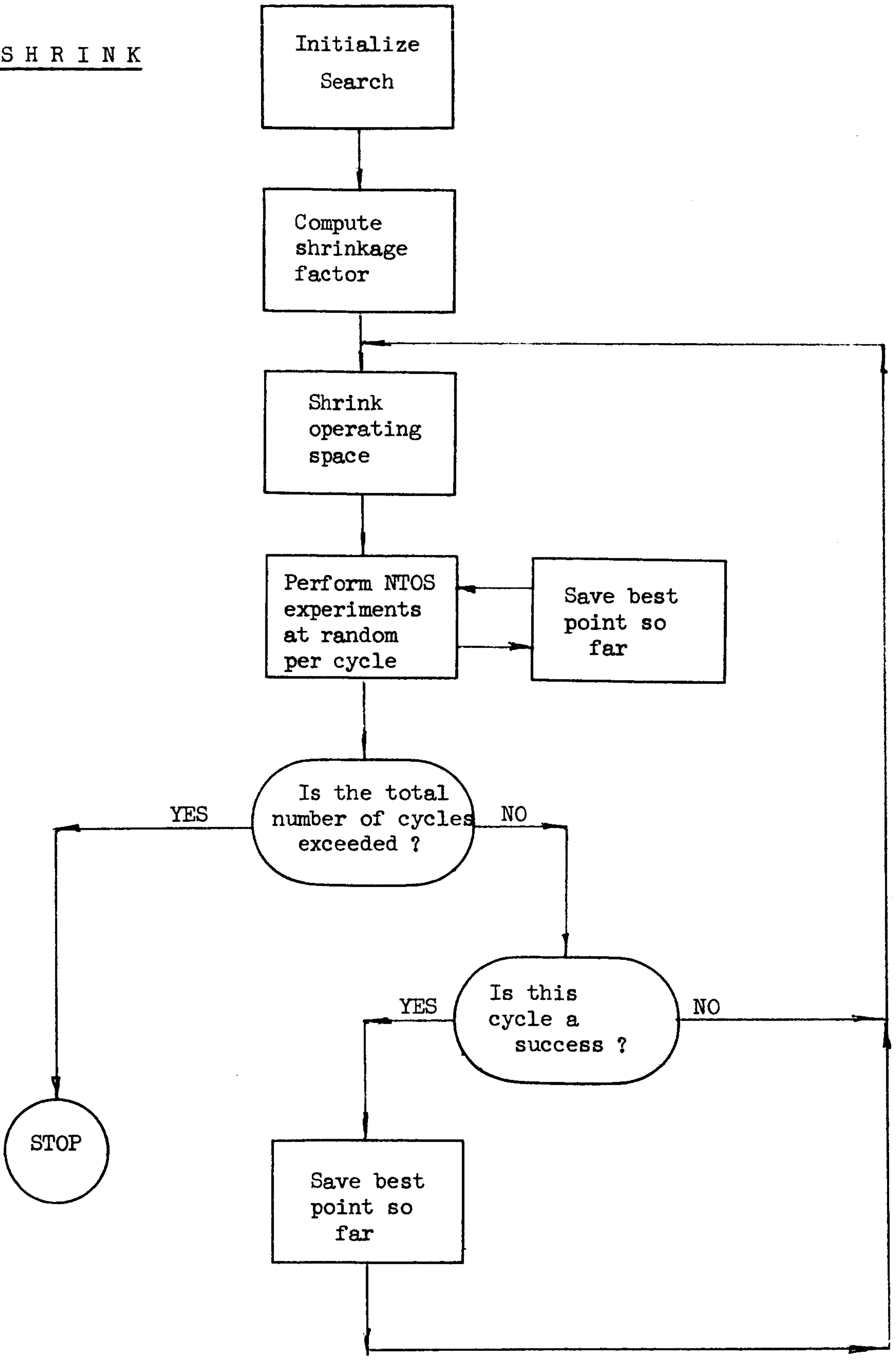


```
      GO TO ( 9,10 ), LL4

 9    LE(I) = 1.0
      LUT(I) = 1.0

      CALL
        EVALT

      FMAX  =  SN
      PFMAX  =  SN
      PBESV(I)  =  PN(I)
      BEVS(I)  =  PN(I)
      ( Initialization )

      CALL
        OUTPUT

10    GO TO ( 1, 2, 3, 4 ), ICODE
 1    CALL CLOCK (IC1)
      CALL RANDOM
      ISH(ICODE) = ISH(ICODE) + 1
      CALL CLOCK (IC2)
      NT(ICODE) = NT(ICODE) + IC2 - IC1
          . . . . . . . . . . .
          . . . . . . . . . . .

20    NLOOK =
      NLOOK + 1
```

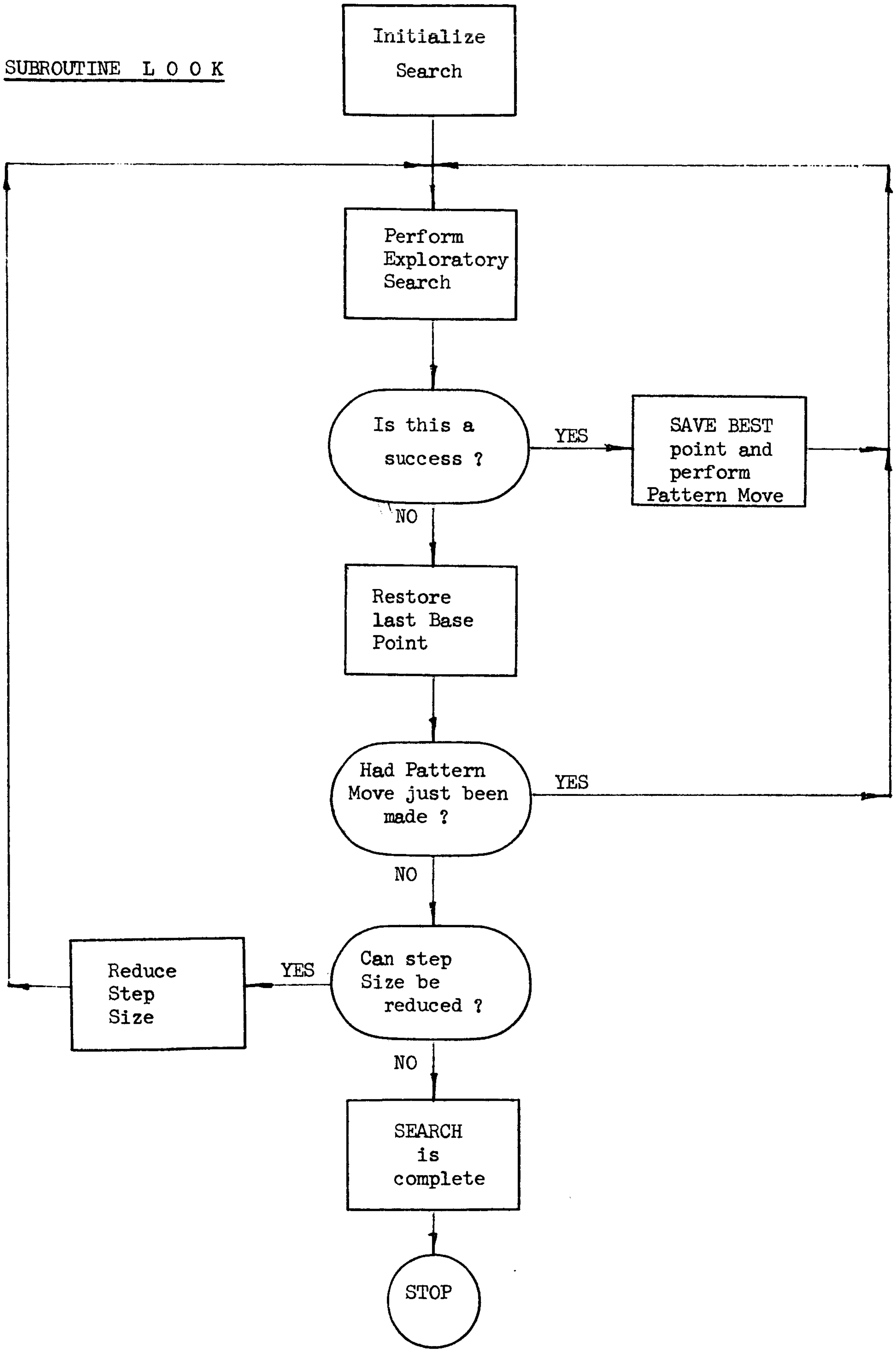
SUBROUTINE R A N D O M



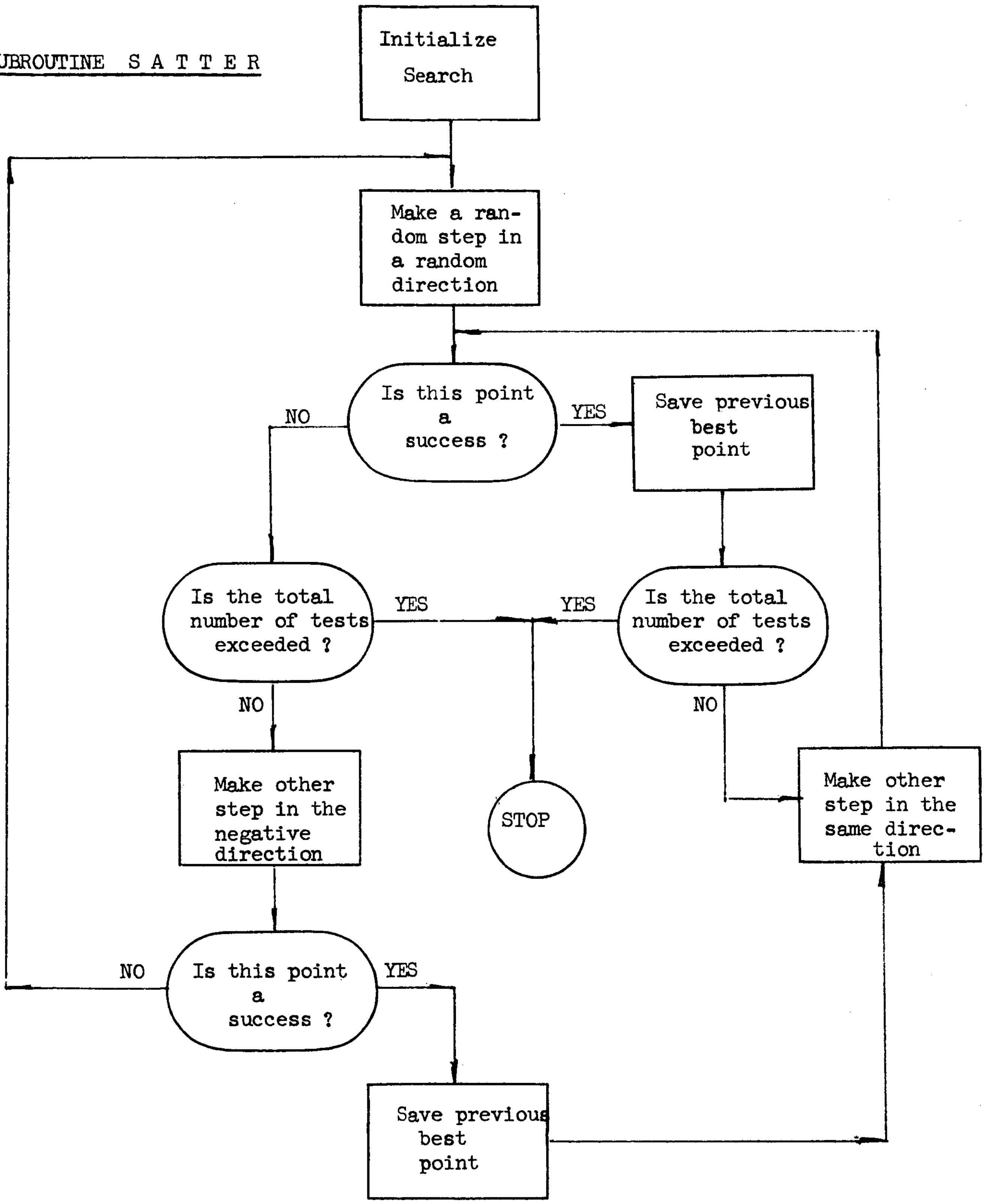
SUBROUTINE S H R I N K



SUBROUTINE L O O K


Initialize Search
Perform Exploratory Search
Is this a success ?
YES
SAVE BEST point and perform Pattern Move
NO
Restore last Base Point
Had Pattern Move just been made ?
YES
NO
Can step Size be reduced ?
YES
Reduce Step Size
NO
SEARCH is complete
STOP

SUBROUTINE S A T T E R


Initialize Search
Make a random step in a random direction
Is this point a success ?
NO
YES
Save previous best point
Is the total number of tests exceeded ?
YES
YES
Is the total number of tests exceeded ?
NO
NO
Make other step in the negative direction
STOP
Make other step in the same direction
NO
Is this point a success ?
YES
Save previous best point

APPENDIX D

FORTRAN IV LISTINGS

```
$JOB 01481,  TIME 003,  PAGES 050,  CARDS 200,  NAME ALBERTO LEON.      GROP 073
$IBFTC FIRST   LIST
C MAIN
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LF(20),LI,KEY(2 ),
     2J1,I1,NFL,LMDS,LIZ(20)
      COMMON/BBC/IC1,IC2,NTIME,NTI10,NLOOK,NT(20),ISH(20)
      COMMON/BBD/A1,B1,C1,A2,B2,C2
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBF/JRES(500),JSTI(500)
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LOI(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBI/IN(3),IN1(3),IN2(3),IN3(3),JN(3),JN1(3),JN2(3),JN3(3)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBK/FORM(12),FORM1(12),FORM2(12),FORM3(12),FORM4(12),FORM5(
     212),FORM6(12),FORM7(12),FORM8(12),FORM9(12),FORM10(12)
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBN/MXTRI,MXNEL,MXTIM
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBP/DEL,DELMIN,DELR,DELTA,SHRUN
      COMMON/BBQ/NTOS,MCYCLO,NTNCOS,SHRON
      COMMON/BBR/NCY,NSF
      COMMON/BBS/KLMN,K1,KRMN
      COMMON/BBT/MRDTN,NDTN,NTIS,SIEP
      COMMON/BBU/MCYS,NIX
      COMMON/BBV/MCYCLE,NTES,NTNCYS
      COMMON/BBW/IS,MI
      COMMON/BBX/DAMBL,PERR,RISIK,THETA,OMEGA,TAU,RO
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NFL1
      LT11=0
      LT22=0
      LT33=0
      CALL CLOCK(LT11)
      NP=1
      ITRIA=1
4     NTIME=0
      LOR=1
      CALL CLOCK(NTIME)
      K6=-3
      K7=-1
      NEL=0
      NEL1=0
      IROUT=1
      IBEST=0
      NLOOK=0
      IC1=0
      IC2=0
      NTI10=0
      NSNIC=0
      DO 50 I=1,20
      NT(I)=0
50    ISH(I)=0
      WRITE(K6,100)
      WRITE(K6,114)NP
C READ AND WRITE REFERENCE SUBROUTINES
```

```
      CALL RWSUB
C READ GENERAL PROGRAM AND OPTIMIZING CODES DATA
      CALL RSDTA
C CALL THE EXECUTIVE SUBROUTINE
121   CALL ADAPT
C SHALL WE CALL SNICON
      IF(KR16.LE.0) GO TO 1
2     CALL SNICON
      NSNIC=NSNIC+1
      LT=LT
      GO TO (1,121), LT
1     NP=NP+1
C PRINT FINAL RESULTS
      WRITE(K6,101)
      WRITE(K6,103)PFMAX,(PRESV(I),I=1,N)
      IF(KR13.LE.0) GO TO 21
C PRINT RESIDUALS IN LINEAR SYSTEMS
      WRITE(K6,115)(TEMPE(I),I=1,N)
21    IF(IROUT.GT.1) GO TO 10
      WRITE(K6,116)
      GO TO 20
10    IROUT=IROUT-1
      WRITE(K6,104)IBEST,IROUT
20    WRITE(K6,108)NLOOK
      WRITE (K6,106)((ISH(I),I),I=1,NCODS)
      ITRIL=ITRIA-1
      IF(KR19.LE.0) GO TO 22
      WRITE(K6,110)(PROB(I),I=1,NCODS)
      WRITE(K6,111)(JSTI(I),I=1,ITRIL)
      WRITE(K6,112)(JRES(I),I=1,ITRIL)
22    CALL CLOCK(NTI10)
      NTIME=NTI10-NTIME
      WRITE (K6,102)NTIME,NFL1
      WRITE (K6,105) ((NT(I),I),I=1,NCODS)
      WRITE(K6,113)
C DO WE HAVE MORE PROBLEMS IN THIS RUN
      IF(NP.LE.KR8) GO TO 4
      WRITE(K6,109)(IN(I),I=1,3),(IN1(I),I=1,3),(IN2(I),I=1,3),(IN3(I),I
     2=1,3)
      WRITE(K6,110)(PROB(I),I=1,NCODS)
      WRITE(K6,111)(JSTI(I),I=1,ITRIL)
      WRITE(K6,112)(JRES(I),I=1,ITRIL)
      CALL CLOCK(LT22)
      LT33=LT22 LT11
      NPPP=NP-1
      WRITE(K6,117)LT33,NPPP
      STOP
 100  FORMAT(1H0,44X,44H U N I V E R S A L  A D A P T I V E  C O D E)
101   FORMAT(34H0 WITH THE FOLLOWING FINAL RESULTS)
102    FORMAT (21H0 TOTAL ELAPSED TIME I10,6H WITH I8,23H FUNCTIONAL EVAL
     2UATIONS)
103    FORMAT (24H0 BEST FUNCTIONAL VALUE E13.6/38H0 WITH THE FOLLOWING O
     2PTIMIZING VECTOR/(1H0 10E13.6))
104    FORMAT (37H0 OPTIMUM POINT WAS REACHED AT TRIAL I5,21H WITH ROUTIN
     2E NUMBER I5)
105    FORMAT (13H0 TOTAL TIME I10,19H IN ROUTINE NUMBER I5)
106    FORMAT (19H0 NUMBER OF TIMES= I5,19H IN ROUTINE NUMBER I5)
108    FORMAT (34H0 OPTIMIZING ROUTINES WERE CALLED I5,18H TIMES AS FOLLO
     2WS=)
109    FORMAT (1H0,4H IN=3(2X,I6)/5H IN1=3(2X,I6)/5H IN2=3(2X,I6)/5H IN3=
      23(2X,I6))
110   FORMAT (32H0 THE FINAL STATE PARAMETERS ARE/(2H  10E13.6))
111   FORMAT (17H0 LIST OF STIMULI/(12(1H ,5I1)))
112   FORMAT (19H0 LIST OF RESPONSES/(12(1H ,5I1)))
113   FORMAT (1H1)
114   FORMAT (25H0 THIS IS PROBLEM NUMBER I4)
115   FORMAT (19H0 THE RESIDUALS ARE/(1H0 10E13.6))
116   FORMAT (36H0 THE INITIAL VALUE WAS NOT IMPROVED)
117   FORMAT (21H0 TOTAL ELAPSED TIME I10,9H FOR THE I4,9H PROBLEMS)
      END
```

```
$IBFTC RSDTA   LIST
C RSDTA
      SUBROUTINE RSDTA
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBC/IC1,IC2,NTIME,NTI10,NLOOK,NT(20),ISH(20)
      COMMON/BBD/A1,B1,C1,A2,B2,C2
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBF/JRES(500),JSTI(500)
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBI/IN(3),IN1(3),IN2(3),IN3(3),JN(3),JN1(3),JN2(3),JN3(3)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBK/FORM(12),FORM1(12),FORM2(12),FORM3(12),FORM4(12),FORM5(
     212),FORM6(12),FORM7(12),FORM8(12),FORM9(12),FORM10(12)
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBN/MXTRI,MXNEL,MXTIM
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBP/DEL,DELMIN,DELR,DELTA,SHRUN
      COMMON/BBQ/NTOS,MCYCLO,NTNCOS,SHRON
      COMMON/BBR/NCY,NSE
      COMMON/BBS/KLMN,K1,KRMN
      COMMON/BBT/MRDTN,NDTN,NTIS,STEP
      COMMON/BBU/MCYS,NIX
      COMMON/BBV/MCYCLE,NTES,NTNCYS
      COMMON/BBW/IS,MI
      COMMON/BBX/DAMBL,PERR,RISTK,THETA,OMEGA,TAU,RO
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      READ(K7,100)KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20,KR21,KR22,KR23,KR24,KR25
C SHOULD WE READ FORMATS
      IF(KR15.LE.0) GO TO 3
4     CALL RFMTS
3     READ(K7,100)N,M,NC,NTSNC,NTC
      READ(K7,100)(NORVI(I),I=1,N)
      READ(K7,100)MXTRI,NDECI,MXNEL,MXTIM,LMDS,LL1,LL2,LL3,LL4,LL5,LL6,
     2LOR
      READ(K7,101)EP1,TOL,TOLS
      READ(K7,103)(IN(I),I=1,3)
      READ(K7,103)(IN1(I),I=1,3)
      READ(K7,103)(IN2(I),I=1,3)
      READ(K7,103)(IN3(I),I=1,3)
      READ(K7,101)(PN(I),I=1,N)
      READ(K7,101)(PNMAX(I),I=1,N)
      READ(K7,101)(PNMIN(I),I=1,N)
      READ(K7,100)NCODS
      READ(K7,100)MCYCLE,NTES,NTNCYS
      READ(K7,102)MCYCLO,NTOS,NTNCOS,SHRON
      READ(K7,102)MRDTN,NTIS,NDTN,STEP
      READ(K7,101)DEL,DELR,DELMIN,DELTA,SHRUN
      READ(K7,101)DAMBL,THETA,OMEGA,TAU,RO,RISTK
C DO WE PRINT PROGRAM PARAMETERS
      IF(KR14.GT.0) GO TO 2
      WRITE(K6,104)
      WRITE(K6,119)KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR1
     22,KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      WRITE(K6,105)N,M,NC,NTSNC,NTC
      WRITE(K6,106)(NORVI(I),I=1,N)
      WRITE(K6,107)MXTRI,NDECI,MXNEL,MXTIM,LMDS
      WRITE(K6,108)EP1,TOL,TOLS
      WRITE(K6,109)(IN(I),I=1,3)
      WRITE(K6,110)(IN1(I),I=1,3)
      WRITE(K6,111)(IN2(I),I=1,3)
      WRITE(K6,112)(IN3(I),I=1,3)
      WRITE(K6,113)(PN(I),I=1,N)
      WRITE(K6,114)MCYCLE,NTES,NTNCYS
      WRITE(K6,115)MCYCLO,NTOS,NTNCOS,SHRON
      WRITE(K6,116)MRDTN,NTIS,NDTN,STEP
      WRITE(K6,117)DEL,DELR,DELMIN,DELTA,SHRUN
      WRITE(K6,118)DAMBL,THETA,OMEGA,TAU,RO,RISTK
100   FORMAT(24I3)
101   FORMAT(5E13.6)
102   FORMAT(3I3,E13.6)
103   FORMAT(3I6)
104   FORMAT (27H0 PARAMETERS OF THE PROGRAM)
105   FORMAT (5H  N= I4,2X,4H M= I4,2X,5H NC= I4,2X,8H NTSNC= I4,2X,6H N
     2TC= I4)
106   FORMAT (31H ORDER OF ANALYSIS ON VARIABLES/(30I4))
107   FORMAT (9H  MXTRI= I5,2X,8H NDECI= I5,2X,8H MXNEL= I5,2X,8H MXTIM=
     2 I5,2X,7H LMDS= I5)
108   FORMAT (7H  EP1= E13.6,2X,6H TOL= E13.6,2X,7H TOLS= E13.6)
109   FORMAT (6H  IN= 3I8)
110   FORMAT (7H  IN1= 3I8)
111   FORMAT (7H  IN2= 3I8)
112   FORMAT (7H  IN3= 3I8)
113   FORMAT (16H  INITIAL VECTOR/(1H ,10E13.6))
114   FORMAT (22H  R A N D O M MCYCLE= I5,2X,7H NTES= I5,2X,9H NTNCYS= I
     25)
115   FORMAT (22H  S H R I N K MCYCLO= I5,2X,7H NTOS= I5,2X,9H NTNCOS= I
     25,2X,8H SHRON= E13.6)
116   FORMAT (21H  S A T T E R MRDTN= I5,2X,7H NTIS= I5,2X,7H NDTN= I5,
     22X,7H STEP= E13.6)
117   FORMAT (15H  L O O K DEL= E13.6,2X,7H DELR= E13.6,2X,9H DELMIN= E1
     23.6,2X,8H DELTA= E13.6,2X,8H SHRUN= E13.6)
118   FORMAT (33H PARAMETERS FOR RELAXATION OPTION/9H  DAMBL= E13.6,2X,
     28H THETA= E13.6,2X,8H OMEGA= E13.6,2X,6H TAU= E13.6/6H  RO= E13.6,
     32X,8H RISTK= E13.6)
119   FORMAT (21H LIST OF CONTROL KEYS/(30I4))
2     RETURN
      END
```

```
$IBFTC IDECI   LIST
C IDECI
      SUBROUTINE IDECI
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(2 ),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBC/IC1,IC2,NTIME,NTI10,NLOOK,NT(20),ISH(20)
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      NDECI=NDECI
      GO TO (1,2,3,4,5,6),NDECI
C DECIDING ON NUMBER OF EVALUATIONS
1     NEL=0
      RETURN
C DECIDING ON TIME AT WORK
2     ITIME=0
      RETURN
3     NDECI=NDECI
      RETURN
4     NDECI=NDECI
      RETURN
5     NDECI=NDECI
      RETURN
6     NDECI=NDECI
      RETURN
      END
```

```
$IBFTC OPTIMU  LIST
C OPTIMU
      SUBROUTINE OPTIMU
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(2 ),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBC/IC1,IC2,NTIME,NTI10,NLOOK,NT(20),ISH(20)
      COMMON/BBF/ITRIA,ICODE,NCODS,NDECI,IBESI,IROUT
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),IEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      LL4=LL4
      GO TO (9,10), LL4
C SET UP VECTORS LE AND LUT
9     DO 53 I=1,NCODS
53    LE(I)=1
      DO 50 I=1,NCODS
50    LUT(I)=0
C INITIAL FUNCTIONAL EVALUATION
      CALL EVALT
      KU=KR18
      KR18=1
      LL4=2
C INITIALIZE PARAMETERS
      FMAX=SN
      PFMAX=SN
      DO 51 I=1,N
      PBESV(I)=PN(I)
51    BESV(I)=PN(I)
      IF(KR12) 10,10,11
11    CALL OUTPUT
10    KR18=KU
      ICODE=ICODE
C CALL PROPER OPTIMIZING CODE
      GO TO (1,2,3,4),ICODE
1     IC1=0
      IC2=0
      CALL CLOCK(IC1)
      CALL RANDOM
      ISH(ICODE)=ISH(ICODE)+1
      CALL CLOCK(IC2)
      NT(ICODE)=NT(ICODE)+IC2-IC1
      GO TO 20
2     IC1=0
      IC2=0
      CALL CLOCK(IC1)
      CALL SHRINK
      CALL CLOCK(IC2)
      ISH(ICODE)=ISH(ICODE)+1
      NT(ICODE)=NT(ICODE)+IC2-IC1
      GO TO 20
3     IC1=0
      IC2=0
      CALL CLOCK(IC1)
      CALL SATTER
      CALL CLOCK(IC2)
      ISH(ICODE)=ISH(ICODE)+1
      NT(ICODE)=NT(ICODE)+IC2-IC1
      GO TO 20
4     IC1=0
      IC2=0
      CALL CLOCK(IC1)
      CALL LOOK
      ISH(ICODE)=ISH(ICODE)+1
      CALL CLOCK(IC2)
      NT(ICODE)=NT(ICODE)+IC2-IC1
20    NLOOK=NLOOK+1
      NGO=NGO
C IF NGO IS ONE TEST DECISION CRITERIA AND CONTINUE PROCESS
C IF NGO IS TWO INTERNAL STOPPING RULE OF OPTIMIZING CODE OCURRED.
      GO TO (21,25),NGO
21    CALL DECIS
      LGO=LGO
C IF LGO IS ONE USE SAME CODE AGAIN
C IF LGO IS TWO TRY SWITCHING CODES
      GO TO (15,14),LGO
15    CALL EVALT
      NEL=NEL+1
      NEL1=NEL1 1
      IF(KR20) 17,16,12
17    LL3=2
16    CALL OUTPUT
12    GO TO 10
25    IF(KR20) 42,41,40
42    LL3=2
41    CALL OUTPUT
40    WRITE(K6,30)ICODE,ITRIA,NEL
      KEY(ICODE)=0
C TRY SWITCHING
14    LOR=1
      DO 54 I=1,NCODS
54    LE(I)=1
      RETURN
30    FORMAT(24H0 STOPPING RULE OF CODE I5,27H HAS BEEN REACHED AT TRIAL
     2 I5,6H WITH I8,23H FUNCTIONAL EVALUATIONS)
      END
```

```
$IBFTC DECIS   LIST
C DECIS
      SUBROUTINE DECIS
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBC/IC1,IC2,NTIME,NTI10,NLOOK,NT(20),ISH(20)
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBN/MXTRI,MXNEL,MXTIM
      NDECI=NDECI
      GO TO (1,2,3,4,5,6),NDECI
C DECIDING ON NUMBER OF EVALUATIONS
1     IF(MXNEL.LT.NEL) GO TO 11
10    LGO=1
      RETURN
C DECIDING ON TIME AT WORK
11    LGO=2
      RETURN
2     IF(MXTIM.LT.ITIME) GO TO 21
20    LGO=1
      RETURN
21    LGO=2
      RETURN
3     NDECI=NDECI
      RETURN
4     NDECI=NDECI
      RETURN
5     NDECI=NDECI
      RETURN
6     NDECI=NDECI
      RETURN
      END
```

```
$IBFTC RCPRO   LIST
C RCPRO
      SUBROUTINE RCPRO
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBD/A1,B1,C1,A2,B2,C2
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBF/JRES(500),JSTI(500)
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      LMDS=LMDS
      GO TO (1,2,3,4,5),LMDS
C FLOOD'S STOCHASTIC LEARNING MODEL
1     LL1=LL1
      GO TO (70,80),LL1
70    ZD1=1.0/FLOAT(NCODS-1)
      ZD2=1.0/FLOAT(NCODS-2)
      AA1=A1-B1
      EE1=(1.0-B1-C1)*ZD2
      CC1=(1.0-A1)*ZD1-EE1
      DD1=C1-EE1
      AA2=A2-B2
      EE2=(1.0-B2-C2)*ZD2
      CC2=(1.0-A2)*ZD1-EE2
      DD2=C2-EE2
      DO 10 I=1,NCODS
10    PROB(I)=SPROB(I)
80    NIT=JSTI(ITRIA)
      JAS=JRES(ITRIA)
      ROS=PROB(JAS)
      IF(NIT-1) 28,22,28
22    TEMP=CC1*ROS+EE1
      W4=DD1
      GO TO 25
28    TEMP=CC2*ROS+EE2
      W4=DD2
25    W1=0.0
      DO 23 J=1,NCODS
      IF(J-JAS) 21,23,21
21    PROB(J)=W4*PROB(J)+TEMP
      W1=W1+PROB(J)
23    CONTINUE
      PROB(JAS)=1.0-W1
      LL1=2
      RETURN
2     LL1=LL1
      RETURN
3     LL1=LL1
      RETURN
4     LL1=LL1
      RETURN
5     LL1=LL1
      RETURN
      END
```

```
$IBFTC RWSUB   LIST
C RWSUB
      SUBROUTINE RWSUB
      COMMON/BBA/K6,K7
3     READ(K7,1)I
      WRITE(K6,1)I
      IF(I)2,3,4
2     CALL SYSTEM
1     FORMAT(1H0,I5,55H0
     2        )
4     RETURN
      END
```

```
$IBFTC RFMTS   LIST
C RFMTS
      SUBROUTINE RFMTS
      COMMON/BBA/K6,K7
      COMMON/BBK/FORM(12),FORM1(12),FORM2(12),FORM3(12),FORM4(12),FORM5(
     212),FORM6(12),FORM7(12),FORM8(12),FORM9(12),FORM10(12)
      READ(K7,1)FORM
      READ(K7,1)FORM1
      READ(K7,1)FORM2
      READ(K7,1)FORM3
      READ(K7,1)FORM4
      READ(K7,1)FORM5
      READ(K7,1)FORM6
      READ(K7,1)FORM7
      READ(K7,1)FORM8
      READ(K7,1)FORM9
      READ(K7,1)FORM10
1     FORMAT(12A6)
      RETURN
      END
```

```
$IBFTC ADAPT   LIST
C ADAPT
      SUBROUTINE ADAPT
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBF/JRES(500),JSTI(500)
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBI/IN(3),IN1(3),IN2(3),IN3(3),JN(3),JN1(3),JN2(3),JN3(3)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBN/MXTRI,MXNEL,MXTIM
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      LL6=LL6
      GO TO (30,31), LL6
C READ PARAMETERS FOR LEARNING MODEL
30    CALL LPRMT
C INITIALIZE TRIALS COUNTING AND LOCK VECTOR
31    I1=0
      LOR=1
      DO 1 J=1,NCODS
      KEY(J)=1
      LIZ(J)=0
1     LOCK(J)=1
7     CALL RAM2A(IN)
C GENERATE NCODS RANDOM NUMBERS
      DO 2 J=1,NCODS
2     RNB(J)=RAM2B(0)
      CALL RAM2C(JN)
      DO 3 J=1,3
3     IN(J)=JN(J)
      J1=1
C IS THE PROBABILITY GREATER THAN THE RANDOM NUMBER
6     IF(PROB(J1).GT.RNB(J1)) GO TO 5
4     J1=J1+1
      IF(NCODS.GE.J1) GO TO 6
      GO TO 7
C ONE OPTIMIZING CODE IS SELECTED
5     ICODE=J1
C WAS THE SELECTED CODE UNSUCCESSFUL IN PREVIOUS TRIALS
      IF(LOCK(ICODE)) 1000,8,9
8     LIZ(ICODE)=LIZ(ICODE)+1
      IF(LIZ(ICODE)-1) 150,160,4
160   I1=I1+1
      IF(NCODS.GT.I1) GO TO 4
10    WRITE(K6,100)
      RETURN
9     I1=0
      DO 40 I=1,NCODS
40    LIZ(I)=0
C BUILD UP VECTOR OF RESPONSES
      JRES(ITRIA)=ICODE
```

```
C INITIALIZE DECISION CRITERIA
      CALL IDECI
C CALL FOR OPTIMIZATION PROCESS
      CALL OPTIMU
C IS THIS TRIAL'S RESULT BETTER THAN PREVIOUS BEST ONE
      IF((FMAX+EP1)-PFMAX) 20,21,21
C TRIAL'S RESULT IS BETTER. BUILD UP STIMULI VECTOR
20    PFMAX=FMAX
      IBEST=ITRIA
      IROUT=ICODE+1
      DO 27 I=1,N
27    PBESV(I)=BESV(I)
      JSTI(ITRIA)=1
      DO 22 J=1,NCODS
22    LOCK(J)=1
      LOCK(ICODE)=KEY(ICODE)
      KEY(ICODE)=1
      WRITE(K6,101)ICODE,ITRIA,NEL
      GO TO 23
C TRIAL'S RESULT IS NOT BETTER
21    JSTI(ITRIA)=0
      LOCK(ICODE)=0
      DO 28 I=1,NCODS
28    KEY(I)=1
      WRITE (K6,102)ICODE,ITRIA,NEL
      IF(KR5) 23,26,25
23    LL3=2
26    CALL OUTPUT
C RECOMPUTE STATE OF THE SYSTEM AT THE END OF TRIAL
25    CALL RCPRO
      ITRIA=ITRIA+1
C IS THE NUMBER OF TRIALS LESS THAN THAN MAXIMUM ALLOWED
      IF(MXTRI.GE.ITRIA) GO TO 1
24    WRITE(K6,103)MXTRI
      RETURN
100   FORMAT(38H0 ALL THE OPTIMIZING CODES F A I L E D)
101   FORMAT(21H0 THE USE OF ROUTINE I5,29H HAS BEEN A SUCCESS IN TRIAL
     2I5 ,6H WITH I8,23H FUNCTIONAL EVALUATIONS)
102   FORMAT(21H0 THE USE OF ROUTINE I5,29H HAS BEEN A FAILURE IN TRIAL
     2I5 ,6H WITH I8,23H FUNCTIONAL EVALUATIONS)
103   FORMAT(24H0 THE MAXIMUM NUMBER OF I5,24H TRIALS HAS BEEN REACHED)
150   WRITE(K6,151)
151   FORMAT (47H0 ERROR.LIZ IS SUPPOSED TO BE GREATER THAN ZERO)
      GO TO 152
1000  WRITE(K6,104)
104   FORMAT (41H0 ERROR.VECTOR LOCK HAS NEGATIVE ELEMENTS)
152   CALL DUMP
      STOP
      END
```

```
$IBFTC LPRMT   LIST
C LPRMT
      SUBROUTINE LPRMT
      COMMON/BBA/K6,K7
      COMMON/BBD/A1,B1,C1,A2,B2,C2
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      READ(K7,101)(SPROB(I),I=1,NCODS)
      READ(K7,101)A1,B1,C1,A2,B2,C2
      DO 1 J=1,NCODS
1     PROB(J)=SPROB(J)
101   FORMAT(5E13.6)
      RETURN
      END
```

```
$IBFTC TOSS    LIST
C TOSS
      SUBROUTINE TOSS
      COMMON/BBA/K6,K7
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBI/IN(3),IN1(3),IN2(3),IN3(3),JN(3),JN1(3),JN2(3),JN3(3)
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      DIMENSION INOR(100)
      CALL RAM2A(IN1)
      J=1
      R1=RAM2B(0)
      R2=N
      IA=R2*R1
      IF(IA)35,20,21
20    IA=IA+1
21    INOR(1)=IA
      DO 4 J=2,N
      R1=RAM2B(0)
      IA=R2*R1
      IF(IA)35,30,31
30    IA=IA+1
31    INOR(J)=IA
      IF(INOR(J)-N)6,6,11
11    INOR(J)=1
6     JS=J-1
5     DO 7 K=1,JS
      IF(INOR(J)-INOR(K))7,9,7
9     INOR(J)=INOR(J)+1
      IF(INOR(J)-N)6,6,10
10    INOR(J)=1
      GO TO 6
7     CONTINUE
4     CONTINUE
      GO TO 100
35    MISS=10000
      WRITE(K6,40)MISS
40    FORMAT(9H0 MISS IS I6)
      CALL ERROR
100   DO 101 J=1,N
      IS=INOR(J)
      NORVI(J)=IS
101   CONTINUE
      CALL RAM2C(JN1)
      DO 102 J=1,3
102   IN1(J)=JN1(J)
      RETURN
      END
```

```
$IBFTC OUTPUT  LIST
C OUTPUT
      SUBROUTINE OUTPUT
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBP/DEL,DELMIN,DELR,DELTA,SHRUN
      COMMON/BBR/NCY,NSE
      COMMON/BBU/MCYS,NIX
      COMMON/BBW/IS,MI
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
C PRINT ACTUAL EVALUATION
      WRITE(K6,100)SN,(PN(KA),KA=1,N)
      WRITE(K6,106)ICODE,NEL,NEL1
C SHOULD WE PRINT PREVIOUS BEST RESULTS
      IF(KR18.GT.0) GO TO 2
      WRITE(K6,101)FMAX,(BESV(KA),KA=1,N)
      WRITE(K6,102)PFMAX,(PBESV(KA),KA=1,N)
2     GO TO (3,4), LL3
4     LF=ICODE
      GO TO (10,11,12,13,14,15), LF
10    WRITE(K6,103)COUNT,IS1
      GO TO 3
11    WRITE(K6,103)COUNT,NCY1
      GO TO 3
12    WRITE(K6,104)COUNT,NIX1
      GO TO 3
13    WRITE(K6,105)COUNT,DEL
      GO TO 3
14    LF=LF
      GO TO 3
15    LF=LF
3     LL3=1
      RETURN
100   FORMAT (25H FUNCTIONAL VALUE IS NOW E13.6/38H WITH INDEPENDENT VAR
     2IABLES AS FOLLOWS/(1H 10E13.6))
101   FORMAT (34H BEST FUNCTIONAL OF PRESENT TRIAL E13.6/38H WITH INDEPE
     2NDENT VARIABLES AS FOLLOWS/(1H 10E13.6))
102   FORMAT (42H BEST FUNCTIONAL VALUE OF PREVIOUS TRIALS E13.6/38H WIT
     2H INDEPENDENT VARIABLES AS FOLLOWS/(1H 10E13.6))
103   FORMAT (34H TOTAL NUMBER OF CYCLES PERFORMED E13.6/26H PROCESS STO
     2PPED IN CYCLE I5)
104   FORMAT (38H TOTAL NUMBER OF CHANGES IN DIRECTION E13.6/39H WITH A
     6TOTAL OF EXPERIMENTS EQUALS TO I5)
105   FORMAT (27H NUMBER OF CYCLES EXECUTED E13.6/17H FINAL STEP SIZE E1
     23.6)
106   FORMAT (14H0 CODE IN USE I4,6H WITH I8,23H FUNCTIONAL EVALUATIONS/
     247H  ACCUMULATED FUNCTIONAL EVALUATIONS UP TO NOW I8)
      END
```

```
$IBFTC SNICON  LIST
C SNICON
      SUBROUTINE SNICON
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBP/DEL,DELMIN,DELR,DELTA,SHRUN
      COMMON/BBX/DAMBL,PERR,RISTK,THETA,OMEGA,TAU,RO
      LL5=LL5
      GO TO (1,2), LL5
C START FROM FARTHEST POINT
1     DO 3 I=1,N
      IF(PBESV(I).LT.0.5*PNMAX(I)) GO TO 12
      PN(I)=PNMIN(I)+0.0005
      GO TO 3
12    PN(I)=PNMAX(I)-0.0005
3     CONTINUE
      IF(NSNIC.LE.NTSNC) GO TO 16
      LT=1
      RETURN
16    WRITE(K6,100)
      LT=2
      LL6=2
      LL4=1
      RETURN
C RELAXATION SCHEME
2     IF(PERR.LE.RISTK) GO TO 21
      DEL=DELMIN
      DELMIN=DELMIN*THETA
      DELTA=DELTA*THETA
      SHRUN=SHRUN*OMEGA
      DAMBL=DAMBL*(PERR/RISTK)*TAU
      TAU=TAU*RO
      IF(NSNIC.GT.NTSNC) GO TO 21
      WRITE(K6,101)
      WRITE(K6,102) DEL,DELMIN,DELTA,SHRUN,DAMBL
      DO 14 J=1,N
14    PN(J)=PBESV(J)
      LT=2
      LL6=2
      LL4=1
      RETURN
21    LT=1
      RETURN
100   FORMAT (50H0 NEW OPTIMIZATION CALCULATION FROM FARTHEST POINT)
101   FORMAT (46H0 PARAMETERS WERE CHANGED AND THE NEW ONES ARE)
102   FORMAT (7H  DEL= E13.6,2X,9H DELMIN= E13.6,2X,8H DELTA= E13.6,2X,
     28H SHRUN= E13.6/28H  NEW RELAXATION PARAMETER= E13.6)
      END
```

```
$IBFTC EVALT   LIST
C EVALT
      SUBROUTINE EVALT
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBK/FORM(12),FORM1(12),FORM2(12),FORM3(12),FORM4(12),FORM5(
     212),FORM6(12),FORM7(12),FORM8(12),FORM9(12),FORM10(12)
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBX/DAMBL,PERR,RISTK,THETA,OMEGA,TAU,RO
      DIMENSION A(50,50),B(50,50),C(100),E(100),PSQ(100),EVECT(100)
      LL2=LL2
      GO TO (1,2), LL2
1     WRITE(K6,FORM)
      DO 6 I=1,N
      DO 6 J=1,N
      A(I,J)=0.0
6     B(I,J)=0.0
      DO 7 J=1,N
      C(J)=0.0
7     E(J)=0.0
      LL2=2
      READ(K7,FORM1)((A(I,J),J=1,N),I=1,M)
      READ(K7,FORM1)((B(I,J),J=1,N),I=1,6)
      READ(K7,FORM1)(C(I),I=1,N)
      READ(K7,FORM1)(E(I),I=1,N)
      IF(KR1.LE.0) GO TO 2
      WRITE(K6,FORM2)
      WRITE(K6,FORM3)
      WRITE(K6,FORM4)
      WRITE(K6,FORM5)(C(I),I=1,6)
      WRITE(K6,FORM6)
      WRITE(K6,FORM5)((B(I,J),J=1,6),I=2,6)
      WRITE(K6,FORM7)
      WRITE(K6,FORM5)(E(I),I=1,6)
      WRITE(K6,FORM8)
      WRITE(K6,FORM5)((A(I,J),J=1,6),I=1,10)
2     RESID=0.0
      SN=0.0
      OBJEC=0.0
      PN(1)=1.0
      DO 5 I=1,M
      TEMPE(I)=0.0
      DO 8 J=1,N
      TEMPE(I)=TEMPE(I)+A(I,J)*PN(J)
8     CONTINUE
      PSQ(I)=TEMPE(I)*TEMPE(I)
5     RESID=RESID+PSQ(I)
      PERR=RESID
      RESID=DAMBL*RESID
      CATS=0.0
      DOGS=0.0
      PIECE=0.0
      DO 3 J=2,N
3     PIECE=PIECE+C(J)*PN(J)
      DO 10 J=2,N
      DO 10 I=2,N
10    DOGS=DOGS+B(I,J)*PN(J)*PN(I)
      DO 9 J=2,N
9     CATS=CATS+E(J)*PN(J)**3
      OBJEC=PIECE+DOGS+CATS
      SN=RESID+OBJEC
      RETURN
      END
```

```
$IBFTC SYSTEM  LIST
C SYSTEM
      SUBROUTINE SYSTEM
      STOP
      END
$IBFTC ERROR   LIST
C ERROR
      SUBROUTINE ERROR
      CALL DUMP
      STOP
      END
$IBFTC CLOCK   LIST
C CLOCK
      SUBROUTINE CLOCK(IDUS)
      CALL TIME (ARCO,IDUS)
      RETURN
      END
$IBFTC TESTCO  LIST
C TESTCO
      SUBROUTINE TESTCO
      RETURN
      END
```

```
$IBFTC LOOK    LIST
C LOOK
      SUBROUTINE LOOK
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBP/DEL,DELMIN,DELR,DELTA,SHRUN
      COMMON/BBS/KLMN,K1,KRMN
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      DIMENSION PC(100),PP(100),PA(100),DP(100),SIGN(100)
      LOR=LOR
      GO TO (250,200), LOR
250   WRITE (K6,100)
100   FORMAT (20H0 L O O K  IS CALLED)
200   LA=LE(4)
      IF(KR17) 252,252,251
251   CALL OUTPUT
252   LA=LA
      GO TO (1,7,11,25,29,33,171,41), LA
C EXPLORATORY SEARCH
1     FMAX=PFMAX
      SP=FMAX
C SC IS THE BASE POINT
      SC=FMAX
      DO 700 KA=1,N
      BESV(KA)=PBESV(KA)
700   PN(KA)=BESV(KA)
      COUNT=0.0
      LOR=2
      NGO=1
      PM=DELTA
      IF(LUT(ICODE).GT.0) GO TO 102
      LUT(ICODE)=1
           DO 2 KA=1,N
           DP(KA)=DEL*(PNMAX(KA)-PNMIN(KA))
2     SIGN(KA)=1.
102        PC(KA)=PN(KA)
           NOS=1
3          K=NORVI(NOS)
C FIND A NEW POINT IN THE POSITIVE DIRECTION OF THE KTH. VARIABLE
           PN(K)=PN(K)+DP(K)*SIGN(K)
C STORE THE NEW POINT IN PA
           PA(K)=PN(K)
      LE(4)=2
      IF(SIGN(K)) 10, 42, 5
5     IF(KR7) 6, 6, 92
92         IF(PN(K)-PNMAX(K))6,4,4
4          PN(K)=PNMAX(K)
6     IF(KR11) 400,400,401
```

```
401    KRMN=1
       CALL TESTCO
       KLMN=K1
       GO TO (400,3,16,22),KLMN
400    RETURN
7            IF(SN-SP) 13, 8,8
C STEP IN THE POSITIVE DIRECTION WAS UNSUCCESSFUL. TRY TWO STEPS IN -
8            PN(K)=PA(K)-2.*DP(K)*SIGN(K)
             PA(K)=PN(K)
       LE(4)=3
       IF(SIGN(K)) 5, 42, 10
10     IF(KR9) 6, 6, 93
93     IF(PNMIN(K)-PN(K)) 6, 9, 9
9            PN(K)=PNMIN(K)
       GO TO 6
11     IF(SN-SP) 125, 12, 12
C POINT IN - TWO STEPS WAS UNSUCCESSFUL. RESTORE THE INITIAL POINT
12           PN(K)=PA(K)+DP(K)*SIGN(K)
             GO TO 14
C BETTER POINT IS FOUND IN - TWO STEPS. KEEP PROCESS GOING
125    SIGN(K)=-SIGN(K)
C BEST VALUE FROM EXPLORATORY SEARCH IS SP
13           SP=SN
       FMAX=SP
       BESV(K)=PN(K)
14           IF(NOS-N) 15, 16, 16
15           NOS=NOS+1
             GO TO 3
C COMPARE BEST EXPLORATORY RESULT (SP) AGAINST BASE POINT (SC)
16           IF(SP-SC+TOL*ABS(SC)) 17,22,22
17     LE(4)=7
             NOS=1
       GO TO 200
C SP IS BETTER THAN BASE POINT SC
171          SC=SP
       FMAX=SC
C TRY A PATTERN MOVE
       DELTA=DELTA*SHRUN
       COUNT=COUNT+1.0
       DO 500 KA=1,N
C STORE OLD BASE POINT IN PP
             PP(KA)=PC(KA)
C ESTABLISH A NEW BASE POINT
             PC(KA)=PN(KA)
       BESV(KA)=PC(KA)
C CALCULATE VARIABLES AFTER PATTERN MOVE
       PN(KA)=PN(KA)*DELTA-PP(KA)
       IF(KR6) 21, 21, 95
95     IF(KR7) 18, 18, 96
96           IF(PN(KA)-PNMAX(KA))600,20,20
600    IF(KR11) 18,18,501
18     IF(KR9) 21, 21, 97
97           IF(PNMIN(KA)-PN(KA))601, 19, 19
601    IF(KR11) 500,500,501
19     PN(KA)=PNMIN(KA)
       GO TO 21
20     PN(KA)=PNMAX(KA)
21     IF(KR11) 500,500,501
501    KRMN=1
       CALL TESTCO
```

```
      KLMN=K1
      GO TO (400,3,16,22,8,24,500,18),KLMN
500   CONTINUE
      IF(KR) 300,300,301
301   CALL TOSS
300        NOS=1
      IF(KR4) 255,255,254
254   CALL OUTPUT
255   LE(4)=4
      RETURN
C POINT AFTER EXPLORATORY SEARCH IS NOT BETTER THAN BASE POINT
C IS THE STEP SIZE SMALLER THAN LIMIT
22         IF(DEL-DELMIN) 23, 23, 24
23    NGO=2
      LE(4)=8
      IF(KR10) 60,61,62
60    LL3=2
61    CALL OUTPUT
62    RETURN
C DECREASE STEP SIZE
24         DEL=DEL*DELR
           DO 241 KA=1,N
241        DP(KA)=DEL*(PNMAX(KA)-PNMIN(KA))
           GO TO 39
C START A NEW EXPLORATORY SEARCH
25    K=NORVI(NOS)
           SP=SN
26         PN(K)=PN(K)+DP(K)*SIGN(K)
           PA(K)=PN(K)
      LE(4)=5
      IF(SIGN(K)) 32, 42, 28
28    IF(KR7) 6, 6, 98
98    IF(PN(K)-PNMAX(K)) 6, 27, 27
27         PN(K)=PNMAX(K)
      GO TO 6
29         IF(SN-SP) 34, 30, 30
C POINT FROM POSITIVE STEP IS NOT BETTER. TRY TWO - STEPS
30         PN(K)=PA(K)-2.*DP(K)*SIGN(K)
           PA(K)=PN(K)
      LE(4)=6
      IF(SIGN(K))28, 42, 32
32    IF(KR9) 6, 6,99
99    IF(PNMIN(K)-PN(K)) 6, 31, 31
31         PN(K)=PNMIN(K)
      GO TO 6
33    IF(SN-SP) 335, 38, 38
C POINT FROM TWO - STEPS IS BETTER. KEEP PROCESS GOING
335   SIGN(K)=-SIGN(K)
C POINT FROM POSITIVE STEP IS BETTER. KEEP PROCESS GOING
34         SP=SN
      FMAX=SP
      BESV(K)=PN(K)
35         IF(NOS-N) 36, 37, 37
36         NOS=NOS+1
      K=NORVI(NOS)
           GO TO 26
37         IF(SP-SC+TOL*ABS(SC)) 17, 39, 39
C POINT FROM TWO - STEPS IS NOT BETTER. RESTORE INITIAL POINT
38         PN(K)=PA(K)+DP(K)*SIGN(K)
```

```
              GO TO 35
 39           SP=SC
          DELTA=PM
              DO 40 KA=1,N
 40           PN(KA)=PC(KA)
          IF(KR) 202,202,201
 201      CALL TOSS
 202      NOS=1
              GO TO 3
 41       GO TO 23
 42       WRITE (K6,101)K,SIGN(K)
 101      FORMAT(22H0 SIGN ERROR. VARIABLE I3,12H HAS SIGN OF   F12.5)
          CALL ERROR
          STOP
          END
```

```
$IBFTC SHRIN   LIST
C SHRINK
      SUBROUTINE SHRINK
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBE/ITRIA,ICODE,NCODS,NDECI,IBEST,IROUT
      COMMON/BBG/LOCK(20),PROB(20),SPROB(20),RNB(20),LUT(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBQ/NTOS,MCYCLO,NTNCOS,SHRON
      COMMON/BBR/NCY,NSE
      COMMON/BBS/KLMN,K1,KRMN
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      DIMENSION PC(100),PP(100),PA(100),DP(100),SIGN(100)
C THIS IS SHRINK RANDOM SEARCH
C NTOS TESTS AT RANDOM
C MCYCLO TOTAL NUMBER OF CYCLES
C NTNCOS  IS THE MAXIMUM  NUMBER OF CYCLES ACCEPTED WITHOUT
C ACCEPTABLE IMPROVEMENT ( CONTINUOUSLY ).
C SHRON IS REDUCTION FACTOR FOR OPERATING SPACE
      LOR=LOR
      GO TO (1,2), LOR
1     WRITE (K6,50)
50    FORMAT (36H0 S H R I N K  R A N D O M IS CALLED)
2     LD=LE(2)
      GO TO (200,24,23), LD
200   LE(2)=2
      COUNT=0.0
      LABOR=0
      LOR=2
      NGO=1
      NCY=0
      NSE=0
      FMAX=PFMAX
      SP=FMAX
      SC=FMAX
      DO 500 MX=1,N
      BESV(MX)=PBESV(MX)
      PN(MX)=BESV(MX)
      PC(MX)=PN(MX)
500   DP(MX)=PN(MX)
      IF(LUT(ICODE).GT.0) GO TO 981
      LUT(ICODE)=1
      P=1.0/FLOAT(N)
      VF=SHRON**P
      VFC=1.0/VF
      DO 30 MX=1,N
      SIGN(MX)=0.5*(PNMAX(MX)-PNMIN(MX))
30    PA(MX)=PNMIN(MX)+SIGN(MX)
981   NCY=NCY+1
C     SHRINK OPERATING SPACE
      VFC=VFC*VF
```

```
C       RUN NTOS TESTS AT RANDOM
982     NSE=NSE+1
C       SELECT LEVELS FOR THE VARIABLES
        DO 87 MX=1,N
100     X=RAM2B(0)
        PP(MX)=VFC*SIGN(MX)*X
        IF(0.5-X) 82,82,83
82      PP(MX)=-PP(MX)
83      PN(MX)=PA(MX)+PP(MX)
C       CHECK TO SEE THAT PN IS WITHIN OPERATING SPACE
        IF(KR6) 21,21,95
95      IF(KR7) 18,18,96
96      IF(PN(MX)-PNMAX(MX)) 18,20,20
18      IF(KR9) 21,21,97
97      IF(PNMIN(MX)-PN(MX)) 21,19,19
19      PN(MX)=PNMIN(MX)
        GO TO 21
20      PN(MX)=PNMAX(MX)
21      IF(KR11) 87,87,501
501     KRMN=1
        CALL TESTCO
        KLMN=K1
        GO TO (87,100),KLMN
87      CONTINUE
        RETURN
24      IF(SC.LT.SN) GO TO 86
C       SC IS THE BEST RESULT SO FAR
91      SC=SN
        FMAX=SC
        DO 93 MX=1,N
        PA(MX)=PN(MX)
93      BESV(KA)=PA(KA)
86      IF(KR17) 98,98,600
600     CALL OUTPUT
98      IF(NTOS.GE.NSE) GO TO 982
        NSE=0
        IF(SC-SP+TOL*ABS(SP)) 17,17,22
17      SP=SC
        FMAX=SP
        LABOR=0
        DO 170 KA=1,N
        PC(KA)=PA(KA)
170     BESV(KA)=PC(KA)
        COUNT=COUNT+1.0
        IF(KR4) 120,120,601
601     CALL OUTPUT
        GO TO 120
22      LABOR=LABOR+1
        IF(NTNCOS.LT.LABOR) GO TO 23
125     COUNT=COUNT+1.0
        IF(KR4) 120,120,126
126     CALL OUTPUT
120     IF(MCYCLO.GE.NCY) GO TO 981
23      NGO=2
        NCY1=NCY
        NCY=0
        NSE=0
        LE(2)=3
        IF(KR10) 25,26,27
```

```
25    LL3=2
26    CALL OUTPUT
27    RETURN
      END
```

```
$IBFTC SAT     LIST
C SATTER
      SUBROUTINE SATTER
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
     -COMMON/BBS/KLMN,K1,KRMN
      COMMON/BBT/MRDTN,NDTN,NTIS,STEP
      COMMON/BBU/MCYS,NIX
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      DIMENSION PC(100),PP(100),PA(100),DP(100),SIGN(100)
C THIS IS SUBROUTINE USING SATTERTHWAITE STRATEGY
C MRDTN RANDOM CHANGES IN DIRECTION
C NDTN MAXIMUM NUMBER OF CHANGES IN DIRECTION ACCEPTED WITHOUT
C IMPROVEMENT ( CONTINUOUSLY ).
C NTIS IS THE TOTAL NUMBER OF EXPERIMENTS TO BE PERFORMED
C STEP IS THE FACTOR FOR STEP SIZE
      LOR=LOR
      GO TO (1,2), LOR
1     WRITE(K6,50)
50    FORMAT (34H0 SATTERTHWAITE STRATEGY IS CALLED)
2     LB=LE(3)
      GO TO (600,5,51,52,1000), LB
600   COUNT=0.0
      MCYS=0
      NIX=0
      NGO=1
      LOR=2
      LABOR=0
      FMAX=PFMAX
      SP=FMAX
      SC=FMAX
      DO 3 MX=1,N
      BESV(MX)=PBESV(MX)
      PN(MX)=BESV(MX)
    3 PC(MX)=PN(MX)
C     TAKE A RANDOM STEP AND RUN A TEST
   13 COUNT=COUNT+1.0
   14 MCYS=MCYS+1
      LE(3)=2
      DO 10 MX=1,N
   60 X=RAM2B(0)
      DP(MX)=STEP *X*(PNMAX(MX)-PNMIN(MX))
      IF(0.5-X) 15,15,25
   15 DP(MX)=-DP(MX)
   25 PN(MX)=PC(MX)+DP(MX)
C     CHECK FOR OUT OF BOUNDS AND SIDE CONDITIONS
      IF(KR6) 21,21,95
   95 IF(KR7) 18,18,96
   96 IF(PN(MX)-PNMAX(MX)) 18,20,20
   18 IF(KR9) 21,21,97
   97 IF(PNMIN(MX)-PN(MX)) 21,19,19
   19 PN(MX)=PNMIN(MX)
      GO TO 21
   20 PN(MX)=PNMAX(MX)
   21 IF(KR11) 10,10,53
   53 KRMN=1
      CALL TESTCO
      KLMN=K1
      GO TO (10,60),KLMN
   10 CONTINUE
      RETURN
5     NIX=NIX+1
      LE(3)=3
      IF(KR17) 30,30,31
   31 CALL OUTPUT
C     IS THE LAST EVALUATION BETTER THAN THE PREVIOUS ONE
   30 IF(SC-SN) 40,300,300
C     LAST EVALUATION IS NOT BETTER THAN PREVIOUS ONE
40    IF(NTIS.LT.NIX) GO TO 1000
C     TRY THE OPPOSITE DIRECTION
   41 DO 210 MX=1,N
      DP(MX)=-DP(MX)
      PN(MX)=PC(MX)+DP(MX)
C     ARE WE STILL WITHIN BOUNDS AND SIDE CONDITIONS
      IF(KR6) 221,221,295
  295 IF(KR7) 218,218,296
  296 IF(PN(MX)-PNMAX(MX)) 218,221,220
  218 IF(KR9) 221,221,297
  297 IF(PNMIN(MX)-PN(MX)) 221,221,219
  219 PN(MX)=PNMIN(MX)+(PNMIN(MX)-PN(MX))
      GO TO 221
  220 PN(MX)=PNMAX(MX)-(PN(MX)-PNMAX(MX))
  221 IF(KR11) 210,210,253
  253 KRMN=1
      CALL TESTCO
      KLMN=K1
      GO TO (210,13), KLMN
  210 CONTINUE
      RETURN
51    NIX=NIX+1
      LE(3)=4
      IF(KR17) 230,230,231
  231 CALL OUTPUT
C     IS THIS EVALUATION BETTER THAN THE PREVIOUS ONE
230   IF(SC.GE.SN)  GO TO 300
290   IF(NTIS.LT.NIX) GO TO 1000
260   IF(MRDTN.LT.MCYS) GO TO 1000
  270 IF(SC-SP+TOL*ABS(SP)) 217,217,222
  217 SP=SC
      FMAX=SP
      DO 57 KA=1,N
57      BESV(KA)=PN(KA)
      LABOR=0
      COUNT=COUNT+1.0
      IF(KR4) 14,14,233
  233 CALL OUTPUT
      GO TO 14
  222 LABOR=LABOR+1
      IF(NDTN.LT.LABOR) GO TO 1000
  225 COUNT=COUNT+1.0
      IF(KR4) 14,14,226
  226 CALL OUTPUT
      GO TO 14
C     THE EVALUATION IS BETTER. TRY ANOTHER STEP IN THE SAME DIRECTION.
C     THE STEP SIZE IS THE SAME AS BEFORE.
  300 SC=SN
      FMAX=SC
      DO 350 MX=1,N
      PC(MX)=PN(MX)
      BESV(MX)=PN(MX)
      PN(MX)=PC(MX)+DP(MX)
C     ARE WE STILL WITHIN BOUNDS AND SIDE CONDITIONS
      IF(KR6) 321,321,395
  395 IF(KR7) 318,318,396
  396 IF(PN(MX)-PNMAX(MX)) 318,321,320
  318 IF(KR9) 321,321,397
  397 IF(PNMIN(MX)-PN(MX)) 321,321,319
  319 PN(MX)=PNMIN(MX)+(PNMIN(MX)-PN(MX))
      GO TO 321
  320 PN(MX)=PNMAX(MX)-(PN(MX)-PNMAX(MX))
  321 IF(KR11) 350,350,302
302   KRMN=1
      CALL TESTCO
      KLMN=K1
      GO TO (350,13), KLMN
  350 CONTINUE
      RETURN
52    NIX=NIX+1
      IF(KR17) 340,340,341
  341 CALL OUTPUT
C     IS THE EVALUATION BETTER THAN THE PREVIOUS ONE
340   IF(SN.GT.SC) GO TO 290
362   IF(NTIS.GE.NIX) GO TO 300
1000  NGO=2
      MCYS=0
      NIX1=NIX
      NIX=0
      LE(3)=5
      IF(KR10) 1006,1010,1011
1006  LL3=2
1010  CALL OUTPUT
1011  RETURN
      END
```

```
$IBFTC RANDO   LIST
C RANDOM
      SUBROUTINE RANDOM
      COMMON/BBA/K6,K7
      COMMON/BBB/LL1,LL2,LL3,LL4,LL5,LL6,LOR,LGO,NGO,LE(20),LT,KEY(20),
     2J1,I1,NEL,LMDS,LIZ(20)
      COMMON/BBH/NORVI(100),PN(100),PNMAX(100),PNMIN(100),BESV(100),TEMP
     2E(100),PBESV(100)
      COMMON/BBJ/KR,KR1,KR2,KR3,KR4,KR5,KR6,KR7,KR8,KR9,KR10,KR11,KR12,
     2KR13,KR14,KR15,KR16,KR17,KR18,KR19,KR20
      COMMON/BBL/COUNT,LABOR
      COMMON/BBM/N,M,NC,NTSNC,NTC,NSNIC
      COMMON/BBO/FMAX,PFMAX,SN
      COMMON/BBS/KLMN,K1,KRMN
      COMMON/BBV/MCYCLE,NTES,NTNCYS
      COMMON/BBW/IS,MI
      COMMON/BBZ/EP1,TOL,TOLS
      COMMON/BCB/NCY1,NIX1,IS1,NEL1
      DIMENSION PC(100),PP(100),PA(100),DP(100),SIGN(100)
C     THIS IS SIMPLE RANDOM SEARCH
C MCYCLE IS THE TOTAL NUMBER OF CYCLES
C NTES IS THE TOTAL NUMBER OF TESTS PER CYCLE
C NTNCYS IS THE MAXIMUM NUMBER OF CYCLES
C WITHOUT ACCEPTABLE IMPROVEMENT ( CONTINUOUSLY ).
      LOR=LOR
      GO TO (1,2), LOR
1     WRITE (K6,50)
50    FORMAT (23H0 R A N D O M IS CALLED)
2     LC=LE(1)
      GO TO (600,51,23), LC
600   LE(1)=2
      COUNT=0.0
      LABOR=0
      LOR=2
      NGO=1
      IS=0
      MI=0
      FMAX=PFMAX
      SP=FMAX
      SC=FMAX
      DO 300 MX=1,N
      BESV(MX)=PBESV(MX)
      PN(MX)=BESV(MX)
      PA(MX)=PN(MX)
300   PC(MX)=PN(MX)
240   IS=IS+1
250   MI=MI+1
      DO 500 MX=1,N
100   X=RAM2B(0)
      PN(MX)=X*(PNMAX(MX)-PNMIN(MX))+PNMIN(MX)
      IF(KR6) 21,21,95
95    IF(KR7) 18,18,96
96    IF(PN(MX)-PNMAX(MX)) 18,20,20
18    IF(KR9) 21,21,97
97    IF(PNMIN(MX)-PN(MX)) 21,19,19
19    PN(MX)=PNMIN(MX)
      GO TO 21
20    PN(MX)=PNMAX(MX)
21    IF(KR11) 500,500,501
```

```
501     KRMN=1
        CALL TESTCO
        KLMN=K1
        GO TO (500,100),KLMN
500     CONTINUE
        RETURN
51      IF(SN.GT.SC) GO TO 220
190     SC=SN
        FMAX=SC
        DO 191 KA=1,N
        PC(KA)=PN(KA)
191     BESV(KA)=PC(KA)
220     IF(KR17) 200,200,225
225     CALL OUTPUT
200     IF(NTES.GE.MI) GO TO 250
        MI=0
        IF(SC-SP+TOL*ABS(SP)) 17,17,22
17      SP=SC
        FMAX=SP
        LABOR=0
        DO 170 KA=1,N
        PA(KA)=PC(KA)
170     BESV(KA)=PA(KA)
        COUNT=COUNT+1.0
        IF(KR4) 210,210,702
702     CALL OUTPUT
        GO TO 210
22      LABOR=LABOR+1
        IF(NTNCYS.LT.LABOR) GO TO 23
125     COUNT=COUNT+1.0
        IF(KR4) 210,210,126
126     CALL OUTPUT
210     IF(MCYCLE.GE.IS) GO TO 240
23      NGO=2
        IS1=IS
        IS=0
        MI=0
        LE(1)=3
        IF(KR10) 25,26,27
25      LL3=2
26      CALL OUTPUT
27      RETURN
        END
```

```
$IBMAP RIEL    LIST
* RANDON NUMBERS UNIFORMLY DISTRIBUTED BETWEEN 0. AND +1.0
       REM RAM2.....UNIFORMLY DISTRIBUTED RANDOM NUMBERS
       REM
       ENTRY   RAM2A
       ENTRY   RAM2B
       ENTRY   RAM2C
       ENTRY   RAM2D
RAM2A  SAVE    1,4
       CLA     3,4
       COM
       ADD ONE                   2 COMP. OF LOCATION OF LAST FIVE
       PAX 0,1                   DIGITS OF INPUT RANDOM NUMBER
       CLA 0,1
       ANA MASK
       TZE *+3
       CLA 0,1
       TRA STORE
       CLA 0,1
       LRS 15
       ADD 1,1
       LRS 15
       ADD 2,1
       LLS 12
STORE  STO FRAM                  RANDOM NUMBER
       RETURN  RAM2A
RAM2B  SAVE
       LDQ FRAM                  MULTIPLY PREVIOUS RANDOM NUMBER
       MPY PF5                   BY 15TH POWER OF 5
       STQ FRAM
       CLA FRAM                  CONVERT TO FLOATING POINT
       ARS 8
       ADD FRAM+2
       FAD FRAM+2
       RETURN  RAM2B
RAM2C  SAVE    1,4
       CLA     3,4
       COM
       ADD ONE                   2 COMP. OF LOCATION OF LAST FIVE
       PAX 0,1                   DIGITS OF INPUT RANDOM NUMBER
       STZ 2,1
       STZ 1,1
       STZ 0,1
       CLA FRAM
       LRS 12
       STD 2,1
       LLS 15
       STD 1,1
       LLS 15
       STD 0,1
       TOV *+2
       RETURN  RAM2C
RAM2D  SAVE
       CLA FRAM
       RETURN  RAM2D
FRAM   DEC 34359738367           ORIGINAL RANDOM NUMBER
PF5    DEC 30517578125           15TH POWER OF 5
       OCT 200000000000
ONE    PZE 1
```

```
MASK    OCT 77777
        END
```

```
$DATA
     0 **  THIS IS RUN NUMBER *  068  *  OF  G R O P E
     0 ** F O R T R A N   IV
     0 $$  REFERENCE LIST OF SUBROUTINES USED IN THE PROGRAM
     0 ** 08/03/64  MAIN  GROPE 066
     0 ** 08/03/64  RSDTA  GROPE 066
     0 ** 08/03/64  IDECI  GROPE 066
     0 ** 08/03/64  OPTIMU  GROPE 066
     0 ** 08/03/64  DECIS  GROPE 066
     0 ** 08/03/64  RCPRO  GROPE 066
     0 ** 08/03/64  ADAPT  GROPE 066
     0 ** 08/03/64  OUTPUT  GROPE 066
     0 ** 08/03/64  LOOK  GROPE 066
     0 ** 08/03/64  SHRINK  GROPE 664
     0 ** 08/03/66  SATTER  GROPE 066
     0 ** 08/03/64  RANDOM  GROPE 066
     0 ** 08/03/64  EVALT  GROPE 066
     0 ** 08/03/64  RWSUB  GROPE 066
     0 ** 08/03/64  RFMTS, LPRMT  GROPE 066
     0 ** 08/03/64  TOSS  GROPE 066
     0 ** 08/03/64  SNICON, SYSTEM, ERROR  GROPE 066
     0 ** 08/03/64  CLOCK, TESTCO  GROPE 066
     0 ** 08/03/64  RAM2 (RAM2A,RAM2B,RAM2C)  GROPE 066
     1 $$  END OF LIST OF SUBROUTINES END REFERENCES
  0  1  0  0  0 -1  1  0  1  1 -1  0  1 -1  0  1  1  0  0  1  1

(1H0,43X,38H SOLUTION OF SHELL DEVELOPMENT PROBLEM)
(8F6.2/8F6.2)
(1H0,50X,29H THE INPUT PROBLEM LOOKS LIKE)
(1H0,48X,32H ********************************)
(1H0,56X,15H E  V E C T O R//)
(1H ,42X,F6.2,4X,F6.2,2X,F6.2,2X,F6.2,2X,F6.2,2X,F6.2)
(1H0,56X,15H C  M A T R I X//)
(1H0,56X,15H D  V E C T O R//)
(1H0,52X,25H B  VECTOR AND  A  MATRIX//)


 16 10 16  4 15
  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16  1
400  1300  0  1  1  1  1  1  2  1  1
 1.000000E-06 1.000000E-04 0.000000E+01
4991361863047185 92
4991361863047185 92


 0.100000E+01 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01
 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01
 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01
 0.000000E+01
 0.300000E+01 0.300000E+01 0.300000E+01 0.300000E+01 0.300000E+01
 0.300000E+01 0.300000E+01 0.300000E+01 0.300000E+01 0.300000E+01
 0.300000E+01 0.300000E+01 0.300000E+01 0.300000E+01 0.300000E+01
 0.300000E+01
 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01
 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01
 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01 0.000000E+01
 0.000000E+01
  4
 80 25  5
```

```
40 50   5 0.080000E+01
50800   5 0.050000E+01
0.025000E+01 0.050000E+01 3.000000E-03 0.200000E+01 0.100000E+01
0.010000E+01 0.010000E+01 0.200000E+01 1.000000E-04 0.100000E+01
1.000000E-04
0.025000E+01 0.025000E+01 0.025000E+01 0.025000E+01
0.099140E+01 0.054230E+01 0.037710E+01 0.076750E+01 0.010090E+01
0.075930E+01
40.00-16.00  2.00   0.00   1.00   0.00  -1.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 2.00   0.00  -2.00   0.00   0.40   2.00   0.00  -1.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.25  -3.50   0.00   2.00   0.00   0.00   0.00   0.00
-1.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 4.00   0.00  -2.00   0.00  -4.00  -1.00   0.00   0.00
 0.00  -1.00   0.00   0.00   0.00   0.00   0.00   0.00
 4.00   0.00  -9.00  -2.00   1.00  -2.80   0.00   0.00
 0.00   0.00  -1.00   0.00   0.00   0.00   0.00   0.00
 1.00   2.00   0.00  -4.00   0.00   0.00   0.00   0.00
 0.00   0.00   0.00  -1.00   0.00   0.00   0.00   0.00
40.00  -1.00  -1.00  -1.00  -1.00  -1.00   0.00   0.00
 0.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00
60.00  -1.00  -2.00  -3.00  -2.00  -1.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00  -1.00   0.00   0.00
-5.00   1.00   2.00   3.00   4.00   5.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00  -1.00   0.00
-1.00   1.00   1.00   1.00   1.00   1.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00  -1.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00  30.00-20.00-10.00  32.00-10.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00-20.00  39.00  -6.00-31.00  32.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00-10.00  -6.00  10.00  -6.00-10.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00  32.00-31.00  -6.00  39.00-20.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00-10.00  32.00-10.00-20.00  30.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00-15.00-27.00-36.00-18.00-12.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
 0.00   4.00   8.00  10.00   6.00   2.00   0.00   0.00
 0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
```

APPENDIX E

OVERLAYING STRUCTURES

STRUCTURE "A"

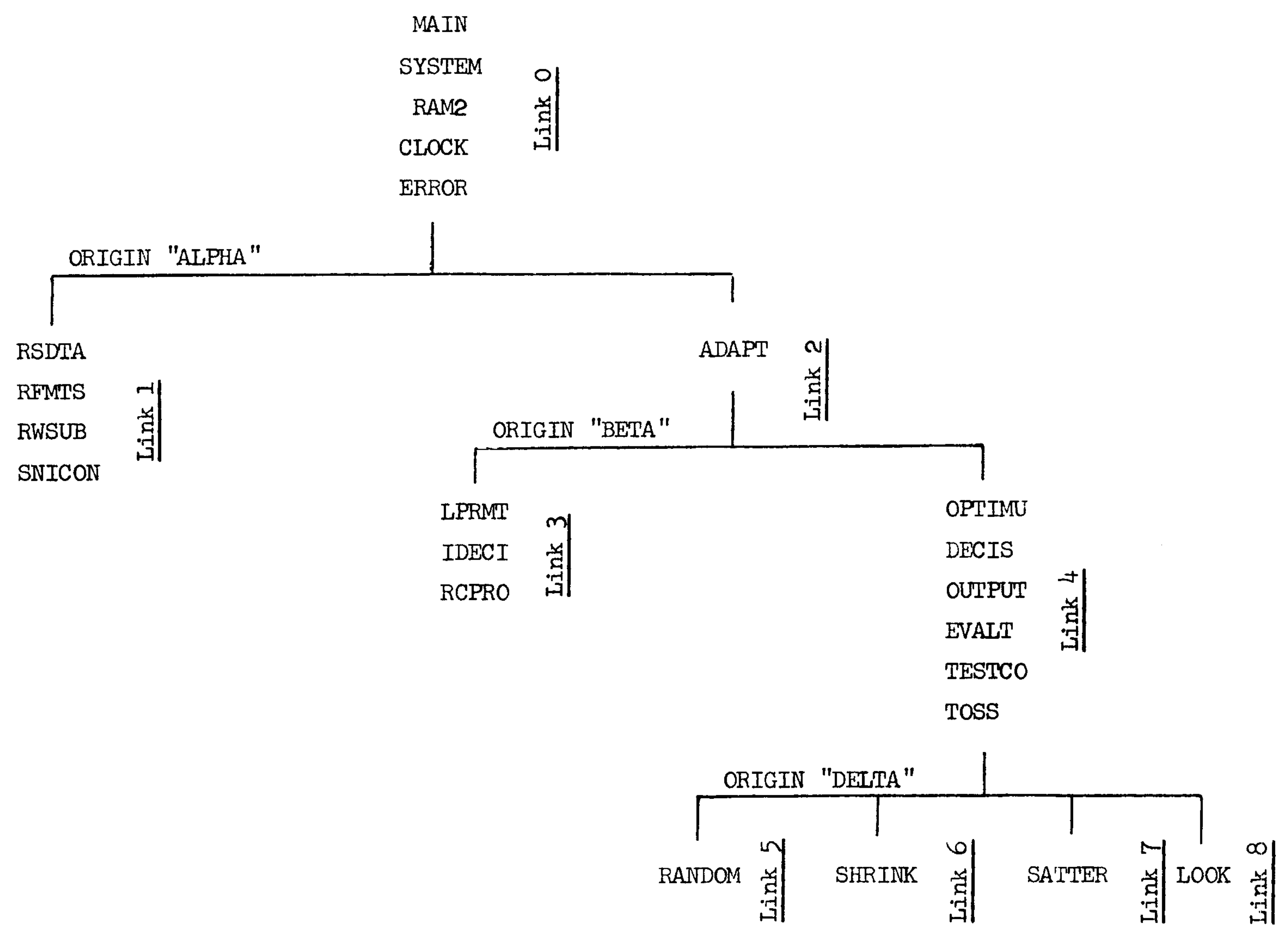


Figure No. 1

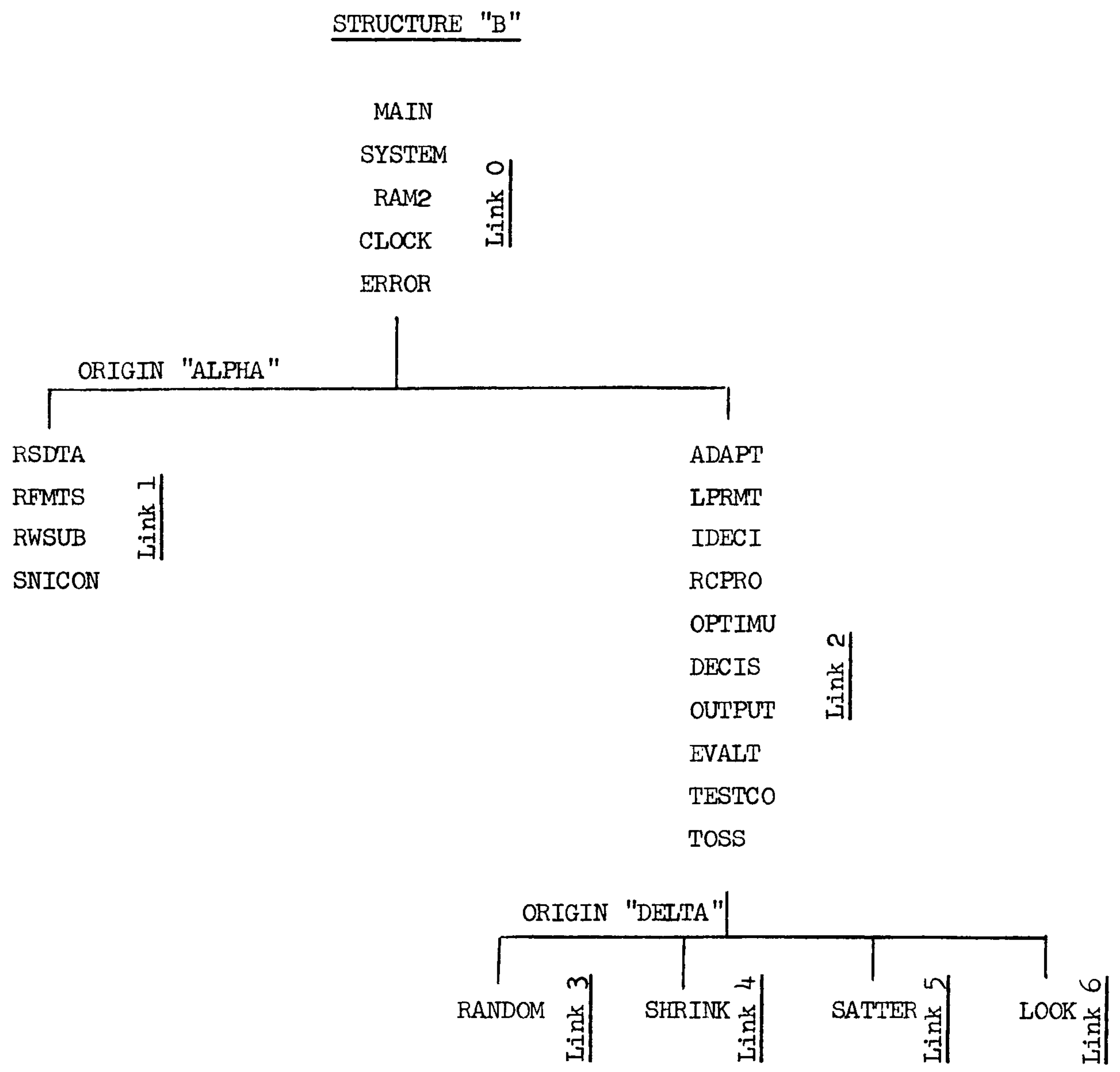

STRUCTURE "B"
MAIN
SYSTEM
RAM2
CLOCK
ERROR
Link 0
ORIGIN "ALPHA"
RSDTA
RFMTS
RWSUB
SNICON
Link 1
ADAPT
LPRMT
IDECI
RCPRO
OPTIMU
DECIS
OUTPUT
EVALT
TESTCO
TOSS
Link 2
ORIGIN "DELTA"
RANDOM
Link 3
SHRINK
Link 4
SATTER
Link 5
LOOK
Link 6

Figure No. 2

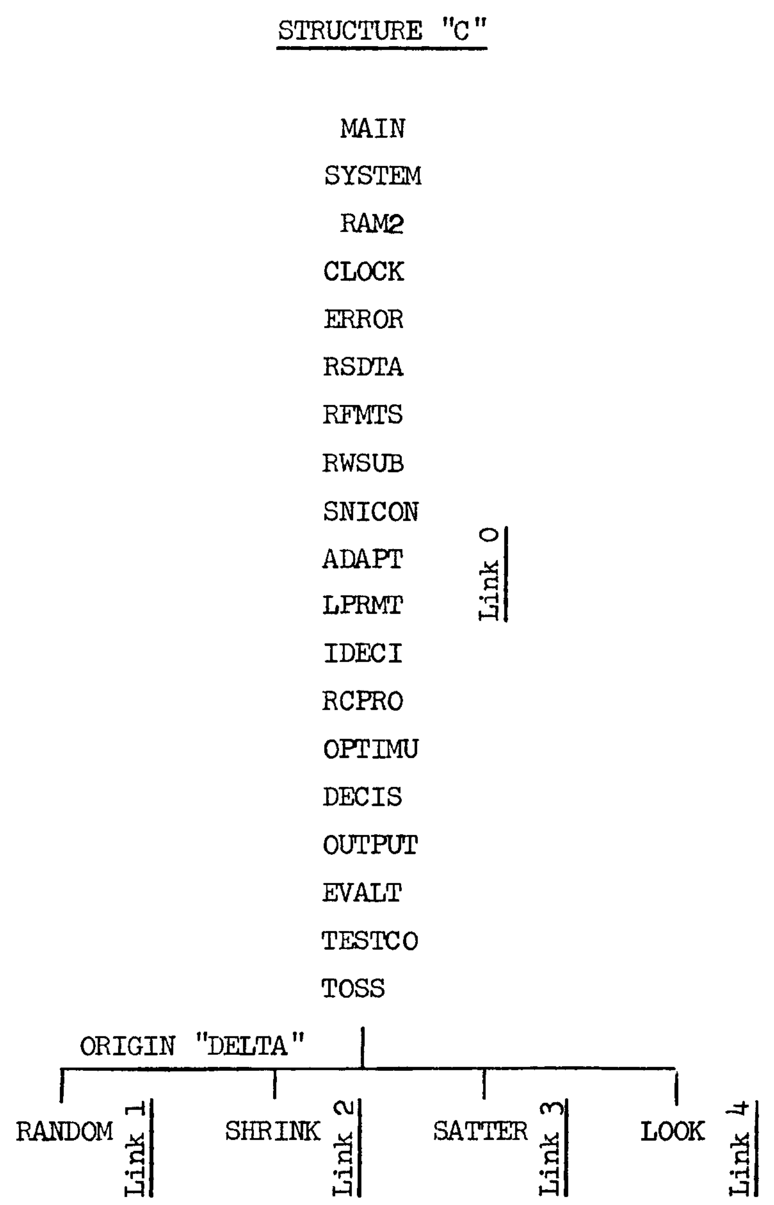

STRUCTURE "C"
MAIN
SYSTEM
RAM2
CLOCK
ERROR
RSDTA
RFMTS
RWSUB
SNICON
ADAPT
LPRMT
IDECI
RCPRO
OPTIMU
DECIS
OUTPUT
EVALT
TESTCO
TOSS
Link 0
ORIGIN "DELTA"
RANDOM
Link 1
SHRINK
Link 2
SATTER
Link 3
LOOK
Link 4

Figure No. 3

OVERLAY STRUCTURE IN FIGURE No.1
ASSIGNED TO CORE STORAGE

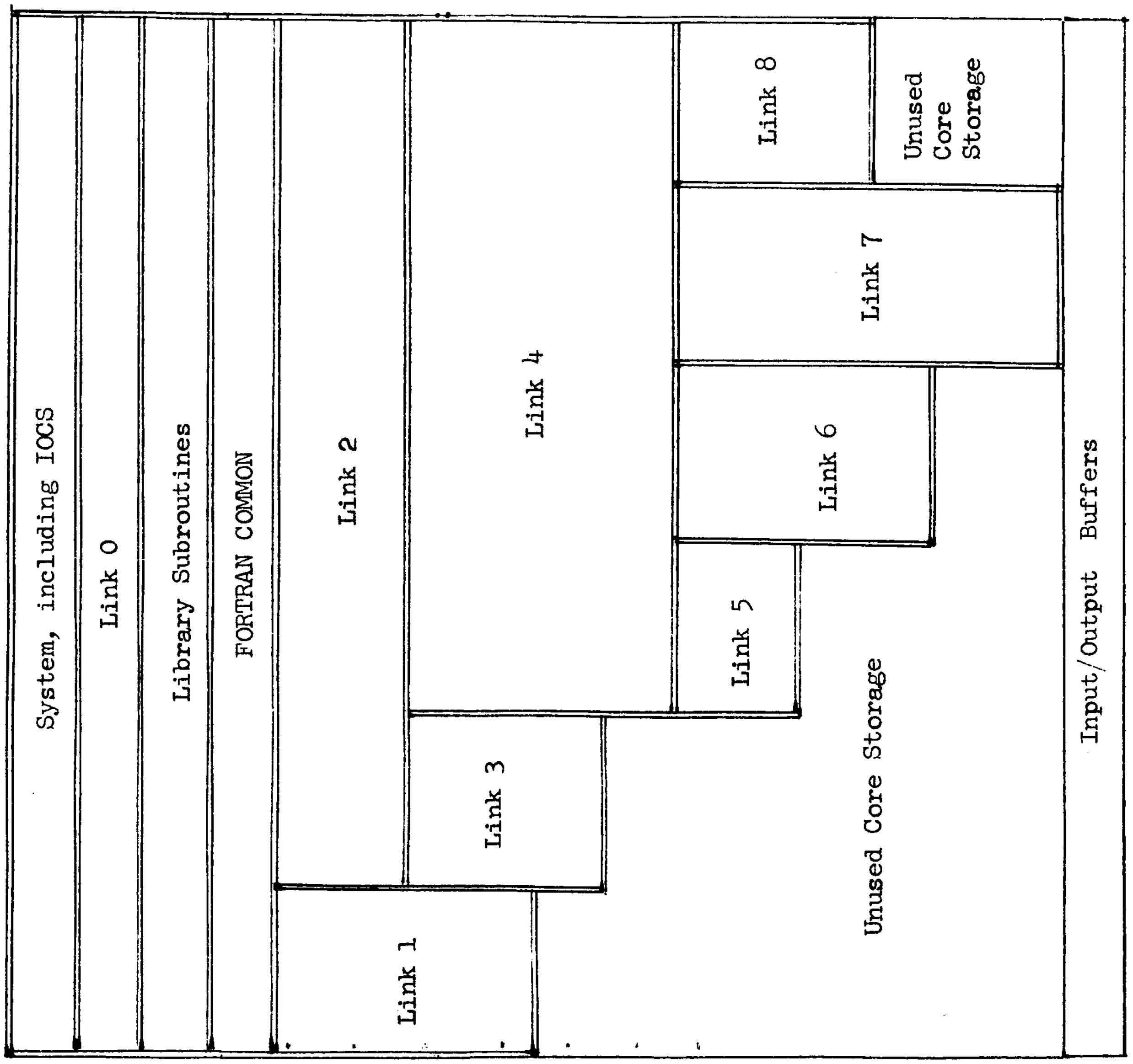


Figure No. 4